VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PODPORA PRO PRÁCI S XML U DATABÁZOVÉHO SERVERU MICROSOFT SQL SERVER 2008

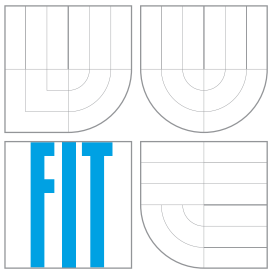
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADKA BÁBÍČKOVÁ

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PODPORA PRO PRÁCI S XML U DATABÁZOVÉHO SERVERU MICROSOFT SQL SERVER 2008

SUPPORT FOR XML IN MICROSOFT SQL SERVER 2008

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. RADKA BÁBÍČKOVÁ
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. JAROSLAV ZENDULKA, CSc.
SUPERVISOR

BRNO 2010

Abstrakt

Diplomová práce je zaměřena na jazyk XML a technologie s ním související. Jazyk XML bezprostředně souvisí s databázemi a jeho podporou na straně databází. V práci je zmapována podpora XML na straně různých databázových produktů a systémů. Podrobněji se věnuje podpoře na straně MS SQL Serveru 2008 od práce s relačními daty, která jsou z databáze exportována do XML formátu a naopak z XML formátu uložena jako relační data až, po XML datový typ a práci s ním prostřednictvím XQuery, přičemž se využívají i různé typy indexování pro zlepšení práce s XML. Podpora u MS SQL Serveru 2008 je demonstrována i výslednou ukázkovou aplikací, která ověřuje teoretické znalosti v praxi.

Abstract

TThis thesis is focused on XML and related technologies. The XML language is directly linked to the databases and its support in databases. The overview of the XML support provided by various database products and systems is presented in this work. Support in the MS SQL Server 2008 is discussed in more detail starting with the mapping of relational data to XML and vice versa to support of the XML data type and work with it through XQuery. Also some indexing techniques are briefly presented. Finally, the support in MS SQL Server 2008 is demonstrated by means of a sample application, which verifies the theoretical knowledge in practice.

Klíčová slova

XML, Oracle, PostgreSQL, MS SQL Server 2008, XML schéma, XML datový typ, XQuery, XSLT, SQLCLR, SQL/XML, LINQ

Keywords

XML, Oracle, PostgreSQL, MS SQL Server 2008, XML schema, XML data type, XQuery, XSLT, SQLCLR, SQL/XML, LINQ

Citace

Radka Bábíčková: Podpora pro práci s XML u databázového serveru Microsoft SQL Server 2008, diplomová práce, Brno, FIT VUT v Brně, 2010

Podpora pro práci s XML u databázového serveru Microsoft SQL Server 2008

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením pana doc., Ing. Jaroslava Zendulku CSc. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Radka Bábíčková
1. června 2010

Poděkování

Ráda bych poděkovala vedoucímu diplomové práce doc. Ing. Jaroslavu Zendulkovi, CSc., za konzultace, cenné rady a připomínky, které mi v průběhu vytváření diplomové práce poskytl, a především za trpělivost, kterou se mnou měl. Dále děkuji všem svým kamarádům, kteří mi poskytli odborné rady ohledně technologií firmy Microsoft, zejména ASP.NET a C#.

Zadání diplomové práce/10382/2009/xbabic02

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2009/2010

Zadání diplomové práce

Řešitel: **Bábíčková Radka, Bc.**
Obor: Informační systémy
Téma: **Podpora pro práci s XML u databázového serveru Microsoft SQL Server 2008**
Support for XML in Microsoft SQL Server 2008
Kategorie: Databáze

Pokyny:

1. Seznamte se rámcově s rozšířením standardu SQL pro práci s XML daty a s podporou u současných databázových SQL serverů (například Oracle 10g, PosgreSQL).
2. Podborně prostudujte podporu pro práci s XML daty v prostředí serveru Microsoft SQL Server 2008.
3. Po dohodě s vedoucím práce navrhněte ukázkovou aplikaci ilustrující použití prostředků pro práci s XML.
4. Navrženou aplikaci realizujte a její funkčnost ověřte na vhodně zvoleném vzorku dat.
5. Diskutujte dosažené výsledky, zejména z pohledu porovnání prostředků dostupných u MS SQL Serveru 2005 a ostatních, se kterými jste se při řešení diplomové práce seznámila.

Literatura:

- Mlýnková, I. a kol.: XML technologie. Grada. 2008. 272 s. ISBN 978-80-247-2725-7.
- Dokumentace k XML. Dostupné na adrese http://www.w3.org/XML/.
- Dokumentace k Microsoft SQL Server 2008. Dostupné na adrese http://www.microsoft.com/sqlserver/2008/en/us/default.aspx a http://msdn.microsoft.com/en-us/sqlserver/default.aspx.
- Dokumentace Oracle na adrese http://www.oracle.com/pls/db102/homepage

Při obhajobě semestrální části diplomového projektu je požadováno:

- Bez požadavků.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Zendulka Jaroslav, doc. Ing., CSc.**, UIFS FIT VUT
Datum zadání: 21. září 2009
Datum odevzdání: 26. května 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Obsah

Kapitola 1

Úvod

Neustále se rozvíjející oblast informačních technologií je spojena s nutností uchovávat data a informace. Honba za neustálým komfortem a lepší prací s dokumenty a daty, které obsahují, vede k nekonečnému vývoji a rozvoji jazyků pro popis a reprezentaci dokumentů. Jedním z těchto jazyků je i značkovací jazyk XML, který umožňuje tvorbu strukturovaných dokumentů a má lepší možnosti pro vyhledávání a práci s informacemi. Byl vytvořen a standardizován konsorciem W3C [22].

S uchováváním dat a informací jsou neodmyslitelně spjaté databáze. V dnešní době téměř všechny databázové systémy podporují více, či méně práci s XML dokumenty. S tímto souvisí i další velká výhoda XML a tou je přenositelnost mezi různými systémy a aplikacemi, tzn. XML je nelicencované a platformově nezávislé.

Jazyk XML je jednoduchý, rychlý a flexibilní textový formát odvozený z SGML (Standard Generalized Markup Language), který je definován ISO 8879 standardem. Jedním z důvodů vzniku XML bylo překonání omezujících prvků jazyka HTML. HTML se neustále rozšiřuje o nové tagy a tím se stává nepřehledným. Naproti tomu XML nemá žádné předdefinované tagy, ale má mnohem přísnější syntaxi. Kombinací jazyka XML a HTML vznikl jazyk XHTML, který se stává následníkem původního HTML.

Jazyk XML se uplatňuje v oblastech práce s dokumenty, s informacemi a v neposlední řadě i v publikování na internetu. Jazyk XML popisuje strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, nezabývá se však vzhledem dokumentu. Vzhled dokumentu se definuje prostřednictvím kaskádových stylů. Dle potřeby lze XML dokument prostřednictvím stylů převést do jiného formátu dokumentu nebo jiné struktury XML dokumentu.

XML dokumenty lze rozdělit na tři skupiny, a to datově orientované, dokumentově orientované a hybridní [1]. Dokumenty zaměřené na data jsou obvykle vytvářeny a zpracovávány aplikacemi, čili plní funkci obálky pro přenos dat nebo manipulaci s nimi.

Dokumentově orientované XML dokumenty jsou vytvářeny a zpracovávány lidmi, čili jsou psány ručně ve formátu XML nebo jsou do něj exportovány. Struktura dat je nepravidelná a málo členitá, vykytují se elementy se smíšeným obsahem a pořadí jednotlivých elementů je zásadní.

S jazykem XML souvisí i celý balík dalších technologií, jako jsou jazyky pro vyhledávání a práci s XML dokumenty a možnost definovat XML schémata. Mezi zmiňované jazyky patří jazyk XPath a XQuery (spojení XPath, SQL, XSLT) [6].

Se vznikem jazyka XML souvisí i vytvoření standardu pro podporu XML v rámci SQL a databázových serverů. Podpora XML je součástí standardu SQL:2003, kdy v rámci tohoto standardu vznikla nová kapitola číslo 14: SQL/XML. Součástí standardu je definice XML datového typu, čtyř operátorů XML (XMLPARSE, XMLSERIALIZE, XMLROOT,

XMLCONCAT), funkce pro generování XML dokumentu z relačních dat (XMLELEMENT, XMLFOREST, XMLATTRIBUTE, XMLNAMESPACES, XMLAGG), definici pravidel pro mapování relačních dat do XML dokumentu, definici namespace deklarace, mapování mezi SQL a XML konstrukcemi (mapování XML jmen do SQL identifikátorů, SQL datových typů do datových typů XML schématu, SQL hodnot do XML, SQL tabulek, schémat a katalogů do XML dokumentu, SQL tabulky do XML).

Podpora XML ze strany jednotlivých databázových systémů (komerčních SŘBD) je veliká a neustále se vyvíjí. Při ukládání XML v relačních databázích lze využít specializovaných schémat pro danou aplikaci, univerzálních schémat bez indexování a univerzálních schémat s indexováním. Při indexování XML dokumentů se využívá mnoho algoritmů a technik relačních SŘBD.

Jedním z mnoha komerčních produktů, který má podporu XML je i Microsoft SQL server 2008. Společnost Microsoft začala s podporou XML u svého databázového systému ve verzi MS SQL Server 2000. V každé nové verzi je tato práce vylepšena a rozšířena o další možnosti. Cílem diplomové práce je do hloubky a detailně popsat podporu a práci s XML v databázovém systému od společnosti Microsoft a následně i demonstrovat ukázkovou aplikací.

Práce je organizována do pěti kapitol.

Druhá kapitola popisuje podporu XML obecně u jednotlivých databázových produktů, jako jsou např. databázové systémy, middleware, nástroje pro dotazování apod. Dále je tu charakterizována podpora u databázového systému firmy Oracle a Microsoft a databázoví systém PostgreSQL .

Obsahem třetí kapitoly je podrobněji rozpracována podpora XML u databázového serveru MS SQL Server 2008. Nedílnou součástí této kapitoly je vývoj a vznik databázového serveru od firmy Microsoft a popis obecných vlastností.

Předposlední kapitola je věnována ukázkové aplikaci. Od jejího návrhu až po samotnou implementaci. Jsou tu skloubeny teoretické znalosti s praktickou částí. Cílem je ověřit si podporu XML ze strany databázového serveru v praxi. Jsou zde popsány použité nástroje k vytvoření aplikace a jednotlivé součásti programu.

Závěrečná kapitola shrnuje celou práci a její výsledky. Jsou zhodnoceny přínosy, případné problémy a nedostatky, které při implementaci vznikly. Nedílnou součástí této kapitoly je i návrh jejího rozšíření a dalšího pokračování.

Kapitola 2

Podpora XML

2.1 XML databázové produkty

V dnešní době existuje již celá řada softwarových produktů s podporou pro práci s XML. Tyto produkty lze rozdělit do několika skupin. Asi nejvýstižnějším rozdělením je rozdělení dle Ronalda Bouretta [2], který je uznávaným odborníkem v oblasti XML a databází. Z důvodu různých pohledů nejsou hranice v tomto rozdělení pevně dány. Některé z produktů nelze jednoznačně zařadit do některé ze skupin nebo jeden produkt spadá typově do více skupin (např. nativní databázový XML systém Tamino je označován i jako XML server). Žádné rozdělení nelze brát striktně, ale i v tomto případě spíše, jako orientační a přehledové.

2.1.1 Databázové systémy s podporou XML

Databázové systémy s podporou XML označované i jako XML enabled jsou databázové systémy, které obsahují rozšíření své funkcionality o přenos dat mezi XML a svou databází. Tyto databázové systémy pracují především s datově orientovanými XML dokumenty. Databáze s podporou XML mohou data poskytovat zpět v jinak strukturovaném XML, než v jakém byly uloženy. Příčinou tohoto jevu jsou převodní algoritmy, jako jsou table based, object based atd. Příkladem těchto databázových systémů jsou Oracle 11g, MS SQL Server 2008, PostgreSQL, IBM DB2, MySQL.

2.1.2 Nativní XML databázové systémy

Nativní XML databázové systémy ukládají XML dokumenty v jejich nativní (přirozené) podobě, včetně jeho logické struktury, poznámek atd. Definice dle Ronalda Bourreta [2]:

"Nativní XML databázový systém je takový systém, který definuje model XML dokumentu a ukládá a získává dokumenty dle tohoto modelu. Model musí obsahovat minimální elementy, atributy, PCDATA a zachovávat dokumentové pořadí prvků. Příkladem takového modelu je datový model XPath, XML Infoset a modely odvozené z DOM a z událostí SAX 1.0."

Jsou vhodné pro práci s XML dokumenty, u kterých záleží na pořadí jednotlivých elementů. Do těchto databází se nejčastěji ukládají dokumentově orientované XML dokumenty a právě tyto dokumenty byly hlavní příčinou pro jejich vznik. Jedním ze zástupců nativních XML databázových systémů je Tamino (fa Software AG), eXist (Wolfgang Meier). Tyto databáze jsou zaměřeny speciálně na ukládání XML dat. Nativní XML databázové systémy jsou určeny pro datově i dokumentově orientované XML dokumenty.

2.1.3 Hybridní XML databázové systémy

Hybridní databázové systémy lze použít buď jako nativní XML databázové systémy nebo jako databázové systémy s podporou XML v závislosti na požadavcích aplikace. Příkladem takového databázového systému je databázový systém Ozone.

2.1.4 Middleware

Pojmem middleware se označuje software, který slouží jako konverzní nebo překladatelská vrstva mezi jednotlivými aplikacemi. Ve funkci prostředníka mezi aplikacemi zajišťuje, jaká data se budou přenášet, jak se budou sdílet a jak zpracovávat. V případě XML databázových produktů je middleware program, který umožňuje transfer dat mezi XML dokumentem a databází. Mezi middleware systémy lze zařadit Microsoft ADO, Allora, Hibernate či IBM DB/XML Transform.

2.1.5 XML servery

XML servery představují aplikační servery, které poskytují svým klientům data ve formátu XML. Příkladem jsou J2EE (Java 2 Platform Enterprise Edition) servery nebo webové servery. Jednoduchým XML serverem by mohl být webový server obsahující ASP (ActiveX Server Pages) stránky, které dle zadaných parametrů v URL načtou data z databáze a vygenerují obsah ve formátu XML.

Představiteli XML serverů jsou Cocoon (Apache Software Fondation), ColdFusion (Adobe), WebObjects (Apple Computer).

2.1.6 Wrappery

Wrappery jsou systémy, které pracují s XML dokumenty jako s relačními daty. Wrapper obalí XML dokument a zpřístupní jeho data v relační formě (v tabulkách). Nad takovými daty lze pak pracovat v jazyce SQL. Mezi představitele wrapperů patří MS SQL Server 2008, Sunopsis XML Driver.

2.1.7 Systémy pro správu obsahu

Systémy pro správu obsahu (CMS - Content Management System) jsou systémy pro ukládání, vyhledávání, správu a publikování dokumentů. Systém pro správu by v našem případě tvořil nadstavbu nad nativním XML databázovým systémem nebo nad systémem souborů pro ukládání XML dokumentů.

Nativní XML databázový systém, nad kterým je CMS vystavěn, je většinou před uživatelem skrytý, uživatel CMS tedy s ním nepřijde přímo do styku. Příkladem CMS systémů jsou CMS (Sorman), Dynabase (Red Bridge Interactive).

2.1.8 Nástroje pro dotazování nad XML

Nástroje pro dotazování nad XML představují samostatné programy, které umožňují provádět operace nad XML dokumenty. V současné době tyto programy využívá relativně mnoho XML databázových jazyků. Mezi nejpoužívanější dotazovací jazyky patří XPath, XQuery, XSQL [6], [24].

Nástroje pro dotazování umí pouze klást dotazy, kdežto nativní XML DBS poskytuje širší funkcionalitu, jako např. ukládání XML dokumentů, transakce. Lze tedy říci, že nativní

XML databázové systémy jsou nadmnožinou nástrojů pro dotazování, protože i nativní XML databázové systémy umožňují dotazování XML dokumentů.

XPath (XML Path Language) definuje stromový model, podle kterého se vyhodnocují všechny výrazy. Stromový model odpovídá logické struktuře daného XML dokumentu. XPath syntaxe je používána pro výběr části XML dokumentu. Výsledkem XPath výrazu může být jeden, více nebo žádný XML element, či atribut nebo může dokonce obsahovat i jiné datové typy.

XQuery (XML Query) je standardizovaný jazyk pro práci s XML soubory, databázemi, webovými stránkami a dokumenty, dokáže získat data i z CGI skriptů a vytvářet XML výsledky vhodné pro zpracování s XSLT. Syntaxe jazyka XQuery je nesourodá a postavená na třech různých prostředcích:

1. výrazy XPath - jazyk XPath je podmnožinou XQuery.
2. tzv. výrazy FLWOR (For, Let, Where, Order by, Return) - jsou inspirované jazykem SQL
3. konstruktory XML

2.1.9 XML data binding nástroje

XML data binding nástroje představují skupinu programových nástrojů, které z XML dokumentu vztváří objekty zapouzdřující data v něm obsažená. Ze schématu XML dokumentu jsou vytvořeny třídy, jejichž instance jsou následně používány pro data konkrétních XML dokumentů. Takto vytvořené objekty jsou přirozenější, lépe charakterizují realitu, aplikační logiku narozdíl od objektů DOM. Mezi XML data binding nástroje lze zařadit .Net Framework nebo JAXP.

2.2 Databázový server Oracle

Mezi databázové servery s podporou XML patří i Oracle Database 11g (dále jen Oracle) [18]. Oracle je představitelem klasické objektově-relační databáze, ve které je použita technologie pro podporu XML. Oracle má schopnosti spravovat nativně veškerá data, od obvyklých firemních informací přes dokumenty XML až po 3D data. K uloženým datům lze přistupovat prostřednictvím vnitřních nástrojů databázového (dále jen DB) systému nebo prostřednictvím uživatelských aplikací.

Oracle podporuje XML datový typ, mapování mezi SQL a XML konstrukcemi, funkcí pro generování XML z relačních dat, generování XML z SQL dat a ANSI SQL:2003 standard, co je součástí SQL/XML standardu, který je obsažen v ISO SQL:2003 standardu.

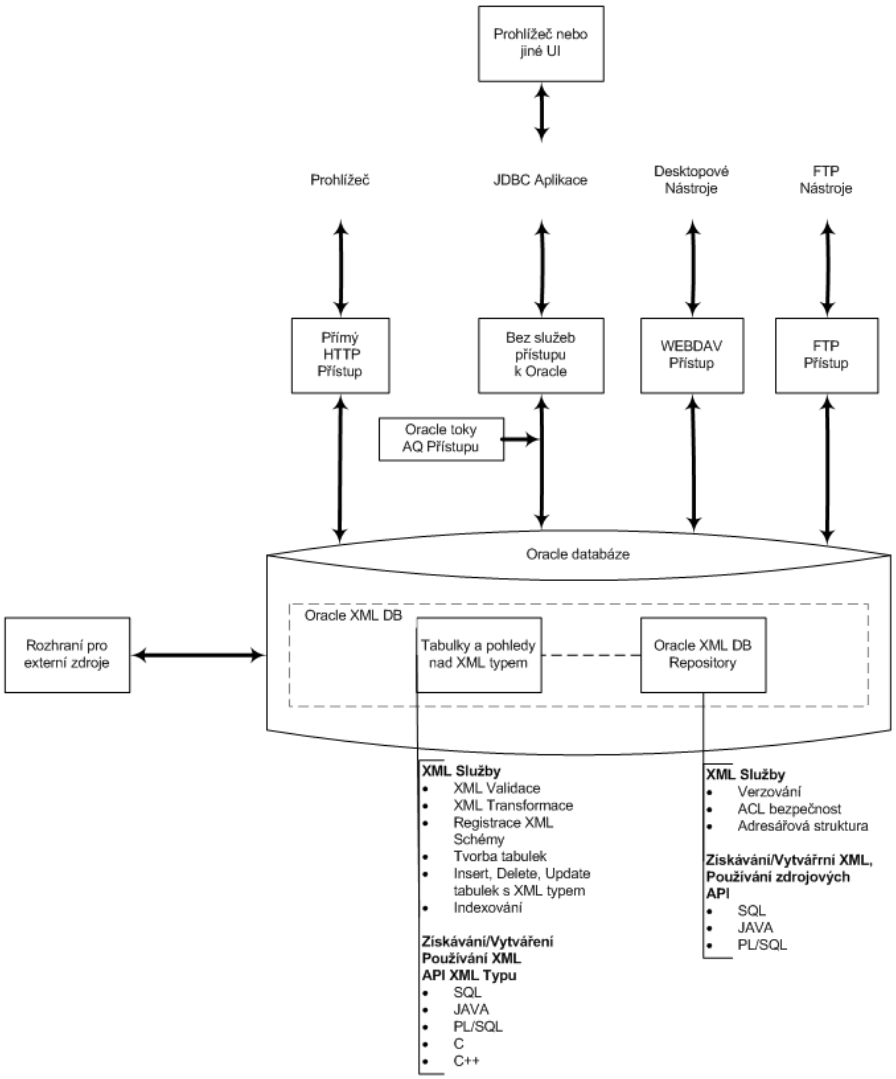
2.2.1 Oracle XML DB

Oracle XML DB představuje rozšiřující balíček prostředků, nástrojů a nových funkčností pro práci s XML daty a dokumenty. Poskytuje plnohodnotnou podporou pro ukládání XML v nativní podobě a podporu pro vyhledávání a dotazování nad těmito daty. Je plně kompatibilní s W3C XML datovým modelem. Oracle XML DB spojuje výhody relačních databází a technologie XML [18].

Jednou z mnoha vlastností je XML/SQL dualita, která umožňuje kombinovat operace XML nad SQL daty a taktéž operace SQL nad XML daty. Dalšími význačnými rysy jsou

nativní datový typ XMLType, dokumentová a DOM přesnost, XML schéma, navigace pomocí XPath, indexy nad XML, nové metody a SQL operátory, XSLT transformace, XML pohledy, generování XML a SQL dotazů, organizace dat (XML DB Repository), správa verzí, spojení s databází (WebDav, HTTP, FTP).

Následující obr. 2.1 popisuje architekturu Oracle XML DB včetně možných komunikačních protokolů, podporovaných XML služeb nebo programovacích a dotazovacích jazyků.



Obrázek 2.1: Architektura Oracle XML DB, upravený z [18]

Balíček Oracle XML DB je složen ze dvou částí, a to balíčku tabulek a pohledů nad XMLType a XML DB Repository.

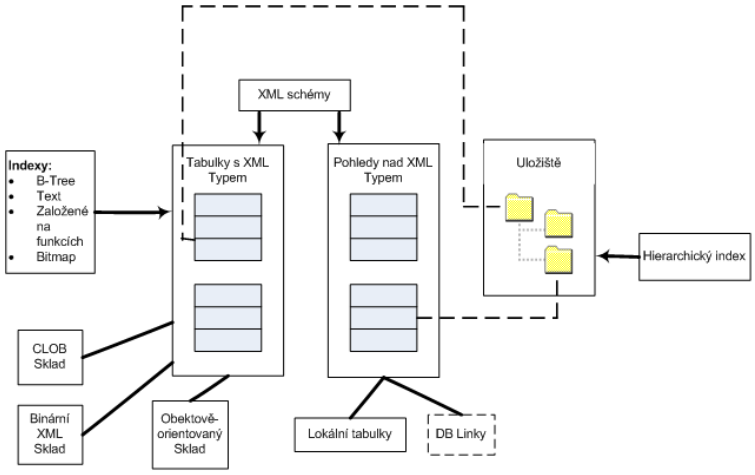
2.2.2 Sklad XMLType

Sklad XMLType poskytuje přirozené uložení XML dat a dokumentů do databáze Oracle, respektivě do databázových tabulek. Cílem XMLType skladu je správa a řízení databázových XMLType tabulek a pohledů, nad kterými zajišťuje uživatelské dotazování, editování nebo

vyhledávání dat.

Při ukládání každého XML dokumentu se vytváří výchozí struktura databázových tabulek. Lze taktéž k danému dokumentu uložit i patřičné XML schéma. Toto XML schéma je zapotřebí nejdříve registrovat pro konkrétní typ XML dokumentu. Když je XML schéma registrováno a struktura daného XML dokumentu odpovídá tomuto schématu, tak jsou data uložena do databáze a XML dokument je automaticky asociován se svým XML schématem, stejně tak jako instance již zaregistrovaného XML schématu.

Následující obr. 2.2, popisuje způsob ukládání XML dokumentů, možné typy indexů používaných nad tabulkami nebo sloupci a způsob propojení tabulek a XML schématu.



Obrázek 2.2: Architektura XMLType Skladu, upravený z [18]

2.2.3 Datový typ XMLType

XMLType je nativním datovým typem pro práci s XML daty. Poskytuje metody vhodné pro operace, jako jsou validace XML sxhématu a XML transformace nad XML daty. XMLType lze použít jako datový typ pro sloupec relační tabulky, typ celé tabulky, typ návratových hodnot funkcí, typ databázových pohledů nebo deklarativní datový typ jazyka PL/SQL.

XMLType přináší mnoho výhod pro práci s XML daty, viz [18], [11] :

- **Podpora XML schématu** - vytváření tabulek a typů daných W3C standardem pro XML schéma, který rozšiřuje SQL DDL
- **XPath vyhledávání a aktualizace dat** - možnost specifikace konkrétní cesty, nad kterou je následně použit SQL dotaz nebo přímá aktualizace hodnoty nalezené jazykem XPath
- **XML indexy** - vytváření indexů pro lepší vyhledávání pomocí jazyka XPath
- **XML operátory** - umožňuje používat nové operátory jako např. XMLTABLE, XMLELEMENT, které zjednodušují vytváření dotazů a generování XML dat z SQL
- **XSLT pro XMLType** - transformace XML dokumentů přes SQL operátor. Výsledkem je XSLT transformace s rychlou odezvou přímo z databáze.

- **Lazy XML přístup** - XMLType poskytuje virtuální DOM. Tento proces optimalizuje využití paměti pouze vkládáním požadovaných řádků dat. Toto umožňuje lepší škálovatelnost při práci s rozsáhlými XML dokumenty. U Oracle balíček DBMS_XMLDOM.

- **XML pohledy** - vytváření agregačních pohledů nad různými částmi XML dokumentů nebo tabulek. Vytváření XML souboru z dat uložených v databázi.

2.2.4 XML DB Repository

Oracle XML DB Repository je optimalizováno pro manipulaci s XML daty. Jedná se o úložiště pro organizaci XML dokumentů, umožňuje komentovat jejich obsah pro snadnější vyhledávání nebo zpracovávat vztahy a závislosti mezi daty v dokumentech.

V Oracle XML DB Repository lze definovat datové položky, jako jsou adresáře nebo soubory s vlastnostmi a parametry:

- jsou definovány přístupovou cestou a jménem.

- obsahují data, která jsou XML daty, ale obsahem nemusí být jenom XML data.

- mají sadu systémově definovaných metadat, jako jsou vlastník a čas vytvoření. Tato metadata se využívají při řízení zdrojů.

- uživatelsky definovaná data, jako jsou komentáře a doplňující parametry, které nejsou součástí obsahu datových položek. Pouze rozšiřují a doplňují informace.

- mají svůj seznam přístupových práv, který určuje, kdo smí přistupovat k položkám a jaké operace se smí nad nimi provádět.

Následující obr. 2.3 představuje základní architektru XML DB Repository a její vnější logickou reprezentaci společně s možnostmi přístupu k datovým položkám a varianty indexování.

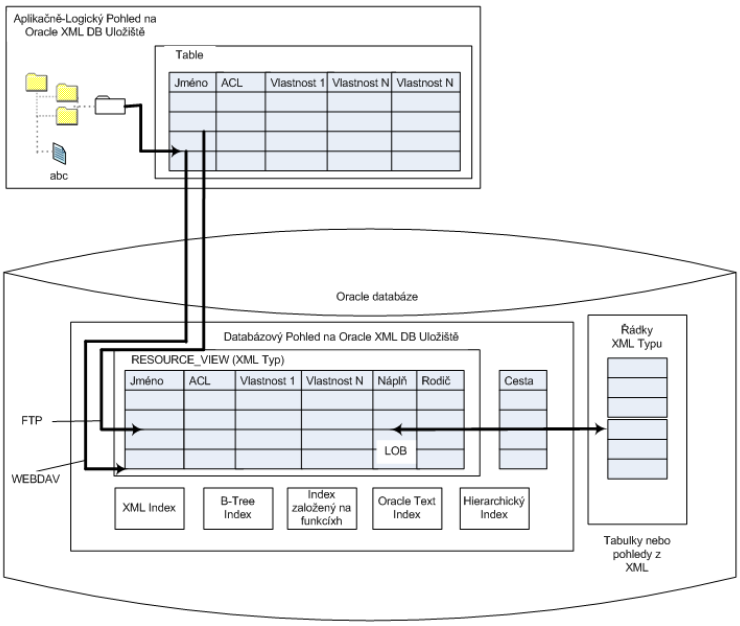
Základní služby XML DB Repository:

- **Služby verzování datových položek** - umožňuje uchovávat data s různou platností. Pro verzování se používá DBMS_XDB_VERSION PL/SQL balíček. Pomocí verzování se uchovává databáze aktuální s možností zálohování dat. Při změně dat se vytvoří nová verze a zároveň je zachována předchozí verze. Podpora verzování je založena na IETF WebDAV standardu.

- **ACL (Access Control List) bezpečnost** - každá datová položka má jasně definovaná přístupová práva. V těchto přístupových právech je definován uživatel, a co může s daty provádět (číst, modifikovat), případně nastavení práv pro provádění příkazu Create, Delete, Update nad datovými položkami. ACL seznam je vnitřní dokument XML. Tento mechanismus práv je založen na WebDAV specifikaci.

- **Správa adresářů a složek** - soubor v XML DB Repository představuje XML data uložena v XMLType tabulce. Soubory mohou být následně rozděleny do jednotlivých adresářů a lze s nimi manipulovat jako u souborového systému.

Možnosti přístupu k datům v XML DB Repository [18]:



Obrázek 2.3: Architektura Oracle XML DB Repository, upravený z [18]

- **Hierarchické indexování** - jeho výhodou je rychlé prohledávání nad datovými položkami (jejich názvy) uloženými v XML DB Repository. Taktéž zajišťuje převod hierarchické souborové reprezentace datových položek na skutečnou vnitřní databázovou podobu.
- **Přístup k datům pomocí SQL** - přístup k datům prostřednictvím RESOURCE_VIEW a PATH_VIEW. RESOURCE_VIEW obsahuje sloupce s datovým typem XMLType, který obsahuje jméno položky, příslušné ACL a jeho vlastnosti. V této tabulce řádek odpovídá jedné datové položce. PATH_VIEW taktéž obsahuje sloupce s XMLType, ale řádek představuje unikátní cestu k položce v XML DB.
- **Přístup k datům prostřednictvím PL/SQL** - přístup k datovým položkám prostřednictvím dotazů jazyka PL/SQL, který obsahuje rozšiřující balíček funkcí pro vytváření, editování a mazání datových položek - DBMS_XDB.
- **Další možnosti přístupu** - přístup k datovým položkám v XML DB Repository prostřednictvím definování cílové cesty URL k databázovému serveru. Využívá se u většiny internetových aplikací. Přístup obsluhuje protokolový server XML DB prostřednictvím protokolů HTTP, HTTPS, FTP nebo WebDAV. Výhodou tohoto přístupu je vytváření adresáře a datových položek přímo v uživatelských aplikacích prostřednictvím SQL nebo PL/SQL. Jinou možností přístupu k datovým položkám představuje jazyk JAVA a Oracle XML DB API pro Javu.

2.2.5 Ukládání dat do databáze Oracle

Je více způsobů jak ukládat data do Oracle databáze, které jsou vhodné v různých situacích. Způsoby ukládání dat do databáze Oracle [18]:

- **Ukládání pomocí relační tabulky bez XMLType** - používal se dříve než vznikla přímá podpora XML dat. Principem je rozklad struktury XML dokumentu na nejmenší elementy, které obsahují konkrétní hodnoty. Každý element je představován jedním sloupcem v relační tabulce. Rozklad probíhá na aplikační úrovni. Využívá se u některých middleware produktů.

- **Ukládání nestrukturovaných dat CLOB** - tento způsob ukládání uchovává přesnou reprezentaci XML dokumentu. Rychlé získávání celého dokumentu, ale operace jen nad částmi dokumentu vyžadují interní dělení, které spotřebovává velké množství paměti a tím způsobuje časovou ztrátu. Nevýhodou je aktualizace informací, při změně musí být změněn celý dokument a ne jen jeho konkrétní část.

- **Objektově-relační reprezentace ukládání** - udržují se přesné informace o struktuře XML dokumentu. Umožňuje ukládání dat v jejich nativní podobě. Dodatečné informace o XML dokumentu jsou uloženy ve speciálních (neviditelných) sloupcích v XML Storage. Existuje i podpora XML schématu a její registrace k databázové struktuře, pak probíhá automatická validace nových informací vůči XML schématu. Vzniklý datový model umožňuje vyhledávat pomocí jazyka XPath v částech dokumentu. Má vysoký DML výkon a zachovává SQL rysy, jako jsou indexy nebo integritní omezení. Nevýhodou je striktní omezení změny formátu dat v XML schématu.

- **Ukládání pomocí PL/SQL balíčku** - má stejný přístup jako objektově-relační varianta, ale pro zápis příkazů vychází z PL/SQL funkcí.

2.3 Databázový server PostgreSQL

PostgreSQL je objektově-relační databázový systém s otevřeným zdrojovým kódem. PostgreSQL je velice stabilní a velice rychlý, má dobrou podporu a integraci pokročilých technologií. Podporuje velkou část standardu SQL a nabízí mnoho moderních funkcí, jako jsou složité dotazy, cizí klíče, triggery, pohledy, transakční integritu, více verzí souběžného přístupu. Nicméně PostgreSQL může být i rozšířen mnoha způsoby samotnými uživateli například přidáním nových datových typů, funkcí, operátorů, agregačních funkcí, indexovacích metod, procedurálními jazyky. Díky liberální licenční politice může být PostgreSQL používán, upravován a distribuován všemi zdarma pro jakékoliv účely, ať už jsou to soukromé, komerční nebo vysokoškolské (GNUPL licence).

Podpora XML v PostgreSQL je teprve na začátku vývoje. Nicméně podporuje XML datový typ, který zahrnuje implementaci funkcí standardu SQL/XML a podporu XPath výrazů.

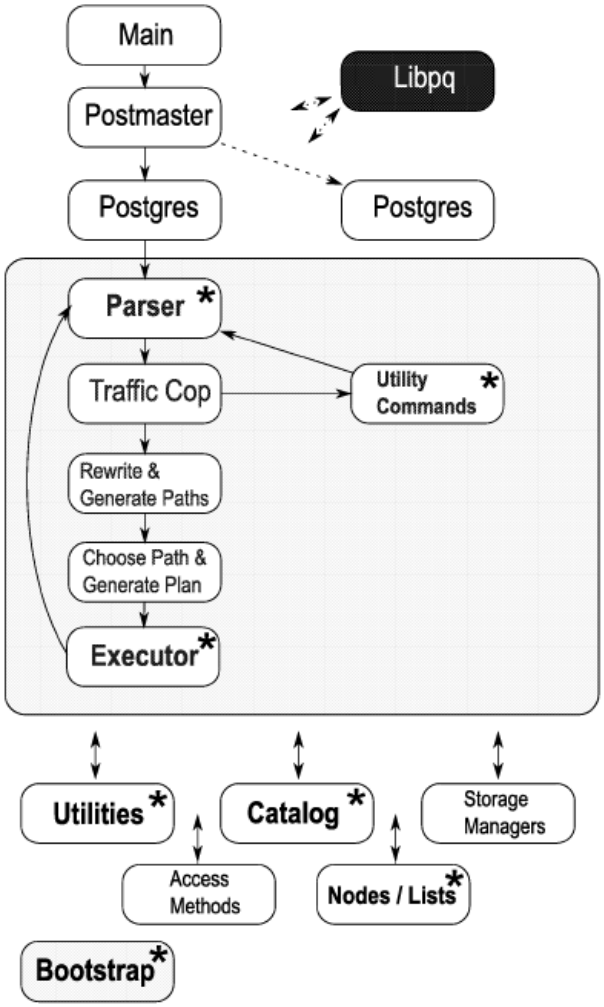
Implementace standardu zahrnuje podporu rutin pro práci s XML daty(XMLPARSE, XMLSERIALIZE) stejně tak pro funkce, prostřednictvím kterých lze transformovat relační data do XML formátu, a výsledkem jsou hodnoty XML typu(XMLELEMENT, XMLCONCAT, XMLCOMMENT). Následně mohou být tyto hodnoty publikovány nebo použity jako hodnoty jiných XML dat. Tato integrace je možná, protože XML typ v PostgreSQL částečně realizuje XQuery data model.
PostgreSQL plně nepodporuje ANSI SQL 2003, každopádně podporuje operace s XML dokumenty přímo v DB, zjednodušené generování XML dokumentů, validace DTD a XML schémy.

2.3.1 Podpora XML datového typu

Podpora nativního XML typu začala implementací souboru základních XML funkcí v modulu contrib/xml2 od Johna Graye. Dnes je již podpora nativního XML typu integrována do jádra databázového systému v souladu s SQL/XML standardem. XML podpora v PostgreSQL byla vyvinuta v souladu s navrženými standardy.

Části sytému, které musely být změněny z důvodu podpory XML typu a SQL/XML funkcí jsou znázorněny pomocí hvězdičky (*) v následujícím obrázku 2.4.

XML datový typ lze používat k ukládání XML dat a je výhodnější oproti ukládání XML dat v textové podobě, protože probíhá kontrola vstupních dat, zda jsou ve správném tvaru. Další nespornou výhodou je podpora funkcí pro bezpečnou práci s těmito daty. PostgreSQL XML typ umožňuje ukládat nejen celý XML dokument ale i jeho části, poskytuje taktéž množinu přípustných hodnot definovaných v XQuery datovém modelu (XDM).



Obrázek 2.4: Vnitřní struktura PostgreSQL. Systémové části, které se změnily pro podporu XML jsou označeny hvězdičkou, převzatý z [9].

2.3.2 Indexování

Pro zvýšení výkonu při práci a vyhledávání v XML dokumentech se využívá indexování a dodatečné struktury. Nejjednodušší typ indexování je funkční B-tree index definovaný prostřednictvím XPath funkcí.

Generalized Search Tree (GiST) je rozšiřující datová struktura, která umožňuje vytvářet indexy nad jakýmikoliv daty a podporuje vyhledávání nad těmito daty. GiST je vyvážená stromová struktura obsahující pár {klíč, ukazatel}, kde klíč je číslo uživatelsky definované třídy.

2.3.3 Full-text vyhledávání

Full-text vyhledávání je technika obecně používaná v dokumentech a XML aplikacích. V PostgreSQL existuje full-text vyhledávací nástroj - tsearch2, který poskytuje širokou a výkonou řadu prostředků jako jsou parsovací techniky založené na ispell slovnících, slovnících synonym, podpory tezauru apod. PostgreSQL full-text vyhledávací prostředky mohou být lehce použity pro potřebnou podporu XML.

2.4 Databázový server MS SQL Server 2008

MS SQL Server je představitelem objektově-relačních databázových systémů s podporou pro práci s XML dokumenty, lze ho taktéž zařadit mezi systémy typu wrapper. MS SQL Server podporuje ukládání dat od relačních až po prostorová data. Součástí MS SQL Serveru jsou služby pro analýzu, dolování dat, integrační služby a reportovací služby. Dále jsou využity technologie pro vysokou dostupnost (minimalizace prostojů), snadnou správu, pro optimalizaci výkonu a lepší škálovatelnost, technologie pro programovatelnost se odvíjí od technologie .NET Framework, v neposlední řadě nabízí vylepšené funkce zabezpečení.

Možnosti XML v MS SQL Serveru 2008, lze rozdělit následovně:

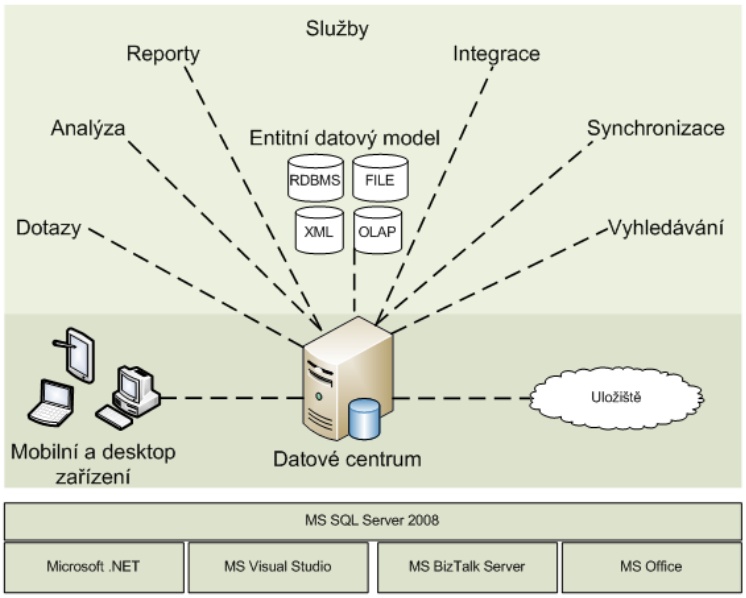
- **Výstup údajů z databázové tabulky v XML formátu** - umožňuje prostřednictvím klauzule FOR XML vypsat relační data jako XML data.
- **Datový typ XML** - ukládání XML dat v jejich nativní podobě. Lze ho používat jako typ sloupce v tabulce, typ parametru uložené procedury, typ návratové hodnoty v uživatelsky definovaných funkcích nebo jako typ proměnné.
- **XML indexování** - indexování XML dat pro lepší vyhledávání a správu těchto dat.
- **XML schéma** - XML data můžou být popsána pomocí schématu. Vytvořené XML schéma lze využít pro validaci při ukládání a modifikaci XML dat.
- **XQuery, XPath, SQL/XML, podpora XML DML** - jazyk SQL sám o sobě nedokáže pracovat s XML daty, proto byly vyvinuty jazyky zaměřené na práci s XML dokumenty, které umožňují vyhledávání v těchto datech, modifikaci a mazání dat.
- **HTTP/SOAP Endpoints** - vlastnost prostřednictvím, které lze využívat webové služby namapované na objekty serveru. Přináší zjednodušení vývoje, nasazování aplikací a jednoduchou konfiguraci podpory webových služeb SQL Serveru.

MS SQL Server plně podporuje standard SQL/XML. Samotná podpora XML je podrobněji popsána v následující kapitole [**?**].

Kapitola 3

MS SQL Server 2008

Firma Microsoft prosazuje princip: Data dostupná kdekoli a kdykoli. S MS SQL Serverem lze získat přístup k datům, i když jsou uložena kdekoli, od serverů v datovém centru přes stolní počítače až po mobilní zařízení. MS SQL Server představuje komplexní platformu, která poskytuje inteligentní přístup k informacím všude, kde je uživatelé potřebují. K datům lze přistupovat i prostřednictvím aplikací, jako jsou MS Office 2007. MS SQL Server umožňuje využívat data z vlastních aplikací vyvinutých na platformě .NET a ve Visual Studiu a taktéž z architektur orientovaných na služby (SOA) a obchodních procesů prostřednictvím BizTalk Serveru. Tento princip je ilustrován na obrázku 3.1.



Obrázek 3.1: Vize společnosti Microsoft pro datovou platformu, upravený z [14]

Historie a vývoj databázového serveru MS SQL Serveru začíná v roce 1987, kdy společnost Microsoft uzavírá dohodu se společností Sybase, která měla již vyvinutý databázový systém pro UNIX, více viz [13].

Rok	Vlastnosti
1988	První beta verze databázového systému s názvem Ashton-Tate/Microsoft SQL Server, který byl určen pro operační systém OS/2. Tato verze byla k dispozici pro vývojáře, kteří chtěli získat náskok v seznámení se s tímto produktem nebo jej měli hodnotit. Databázový server byl součástí balíku NDK (Network Development Kit), který zahrnoval SQL server, Microsoft LAN Manager a OS/2 1.0. a všechny potřebné softwarové komponenty pro vývojáře v jazyce C.
1989	Byla uvolněna verze s názvem SQL server 1.0 pro OS/2. Tato verze byla v podstatě stejná jako Sybase SQL Server 3.0 pro UNIX a VMS.
1990	Byla vydána verze SQL server 1.1, která měla stejné vlastnosti jako verze předchozí, nicméně součástí verze bylo mnoho opravených chyb. Nakonec ale tato verze přeci jenom přinesla novinku a to podporu nové platformy Microsoft Windows 3.0. Tato podpora byla pro SQL server a jeho vývoj zásadní.
1993	Microsoft přišel na trh s SQL serverem 4.2, který byl určen pro Windows NT. Jeho výkon byl konkurenceschopný s dražšími systémy UNIX.
1994	Společnost Microsoft a Sybase oficiálně ukončili své partnerství. Microsft získal výhradní práva k SQL Serveru a Sybase přejmenovali svůj produkt na Adaptive Server Enterprise. Microsoft získal veškerá autorská práva od Sybase.
1995	Vydána nová verze Microsoft SQL Server 6.0 (SQL95), která měla výrazně lepší výkon postavený na replikaci a centralizované správě. Bylo přepsáno jádro technologie.
1996	Uvolněna verze MS SQL Server 6.5, která přinesla vylepšení stávajících technologií a rozšíření v podobě nových funkcí. Microsoft získal osvědčení o tom, že splňují ANSI SQL standard.
1998	Microsoft přišel s verzí MS SQL Server 7.0, kde byl kompletně přepsán na databázový stroj a na první opravdu na GUI založený databázový systém. Součástí systému je velice chytrý správce zámků, který je součásti úplné podpory zamykání na různých úrovních. Byl vytvořen nový dotazovací procesor, který umožnil zpracovávání distribuovaných heterogenních dotazů a AD-hoc dotazů.
2000	V tomto roce byla vydána verze MS SQL Server 2000. Začíná podpora XML ze strany MS SQL Serveru. Mezi další novinky této verze patří triggery, možnost více instancí na jednom počítači, nové datové typy, klauzule REFERENCES, indexy definované nad pohledy a vypočítávanými sloupci. Novinkami v nástrojích jsou přidání Object Browseru do SQL Query Analyseru, možnosti šablon, možnost ladění procedur a samozřejmě vylepšení stávajících nástrojů.
2001 - 2002	Vydána verze MS XML for SQL Server Web Release 1, verze MS SQLXML 2.0 a MS SQLXML 3.0.
2005	Microsoft přichází na trh s novou verzí MS SQL Server 2005, která navazuje na verzi 2000. Představuje integrované řešení pro správu a analýzu dat. Výstup údajů z databázové tabulky v XML formátu, XML datový typ, XML indexování a fulltextové XML vyhledávání, XML schéma, podpora jazyka XQuery.
2008	Poslední vydanou verzí je MS SQL Server 2008. Vylepšení dosavadních funkčností, přidání nových datových typů, nový způsob práce s daty LinQ, nový příkaz MERGE, vylepšena administrace a správa dat, změny v relačním stroji (File Stream, Fulltext apod.), vylepšení Business Intelligence, přepracované reportovací služby a v neposlední řadě vylepšená podpora XML (XPath, XQuery, XML DML apod.).

3.1 Výstup údajů z databázové tabulky v XML formátu

Z relačních dat lze pomocí klauzule FOR XML, která se využívá v příkazu SELECT, získat data ve formátu XML. I v nové verzi MS SQL Serveru 2008 se pokračuje v tradici FOR XML klauzule, včetně následujících vylepšení [3]:

- přidání PATH módu zjednodušuje a zlepšuje pružnost procesu stanovení explicitní struktury XML dat
- XSINIL a ABSENT módy pro prvek ELEMENTS poskytují důslednou kontrolu NULL hodnot ve výsledném XML dokumentu.
- možnosti TYPE pro převod výsledného XML dokumentu na XML datový typ
- podpora vnořených FOR XML dotazů

MS SQL Server 2008 taktéž poskytuje podporu pro FOR XML EXPLICIT klauzuli, která umožňuje zadávat explicitní strukturu výstupních XML dat a OPENXML funkce, která provádí konverzi XML dokumentů na relační data.

3.1.1 Klauzule FOR XML

Klauzule FOR XML byla podporována již ve verzi SQL Server 2000 a je neustále rozšiřována. Klauzule FOR XML poskytuje jednoduchý, výkonný a pružný nástroj pro konverzi relačních dat na XML data.

Syntaxe klauzule FOR XML [7]:

```
SELECT * FROM tabulka FOR XML mode [, XMLDATA] [, ELEMENTS] [, BINARY BASE64]
```

přičemž:

- **mode** - určuje, v jakém tvaru bude výsledek. Možnostmi jsou RAW, AUTO, PATH a EXPLICIT.
- **XMLDATA** - tento parametr určí, zda součástí výsledných XML dat bude i XML schéma.
- **ELEMENTS** - určí, že hodnoty sloupců budou vráceny jako elementy řádku. V opačném případě jsou data vráceny jako atributy. Tento parametr lze použít pouze v režimu AUTO.
- **BINARY BASE64** - binární data jsou převedena na formát base64 kódování.

Každý z konverzních módů má svůj vlastní soubor možností konverze a každý nabízí určitý kompromis mezi jednoduchostí a úrovní kontroly nad strukturou a obsahem výsledného XML dokumentu. Zatímco FOR XML RAW a AUTO se hodí pro rychlé vytváření XML s automaticky geerovanými jmény atributů a elementů, mód PATH nabízí mnohem větší kontrolu a flexibilitu nad konečným XML dokumentem. Režim EXPLICIT také zajišťuje velkou kontrolu nad výsledným XML dokumentem, ale tento režim je již zastaralý a neměl by se používat.

Možnosti použití jednotlivých režimů klauzule FOR XML [3]:

- **RAW** - použitelné pro ad hoc FOR XML dotazy a v případě, kdy neznáme předem strukturu výsledného XML dokumentu. Když se změní zdrojová data, tak se můžou dost významně změnit výsledná XML data.
- **AUTO** - taktéž použitelné pro ad hoc dotazy, když chceme ve výsledném XML dokumentu využít jmen tabulek. Toto je zapotřebí, pokud chceme výsledný XML dokument mapovat zpět do původních sloupců v tabulce.
- **PATH** - je hlavně určen pro striktní definování výsledné XML struktury. Tento mód je výhodnější oproti režimu RAW a AUTO, protože vždy známe strukturu výsledného dokumentu v předstihu, i v případě změny struktury tabulky.
- **EXPLICIT** - metoda pro explicitní definici výsledného XML dokumentu. Používání FOR XML EXPLICIT je mnohem složitější a méně intuitivní než FOR XML PATH.

Každý z FOR XML režimů podporuje celou řadu nastavení, které mohou být použity v klauzuli FOR XML, kde budou odděleny čárkou.

Možnosti nastavení:

- **ROOT** - přidává kořenový uzel do výsledného XML dokumentu se jménem elementu, které si uživatel definuje. Využívá se v případě kdy uživatel potřebuje dobře strukturovaný XML dokument s jedním kořenovým uzlem. V případě, kdy nepotřebujeme přidávat kořenový uzel, tak nečiníme, protože přidání zbytečných uzlů vyžaduje další úložný prostor a může způsobit složitější manipulaci s XML dokumentem. Bez volby ROOT ve FOR XML RAW se vygeneruje pouze fragment XML.
- **TYPE** - vrací výsledek jako instanci XML datového typu. Toto nastavení je užitečné zejména, když je zapotřebí sady FOR XML dotazů nebo přiřadit výsledek do proměnné typu XML nebo ji uchovávat v XML sloupcích.
- **XMLDATA** - zařadí na začátek výsledného XML dokumentu XML Data-Reduced (XDR) schéma.
- **XMLSCHEMA** - vloží na začátek výsledného XML dokumentu XML schéma odpovídající danému dokumentu.
- **ELEMENTS XSINIL** - reprezentuje SQL NULL hodnoty s xsi:nil = "true"atributem ve výsledném XML.
- **ELEMENTS ABSENT** - výchozí atribut. Stanovuje podmínku, že NULL hodnoty budou ve výsledném dokumentu odstraněny.
- **BINARY BASE64** - binární data jsou kódována do BASE64 formátu.

Následující tabulka 3.1 představuje podporu nastavení u jednotlivých režimů klauzule FOR XML:

Tabulka 3.1: Klauzule FOR XML možnosti jednotlivých režimů, převzato z [3]

FOR XML klauzule a možnosti nastavení								
	XMLDATA	*XMLSCHEMA*	*ELEMENTS XSINIL*	*ELEMENTS ABSENT*	*BINARY BASE64*	*TYPE*	*ROOT*	*ELEMENTNAME*
FOR XML RAW	•	•	•	•	•	•	•	•
FOR XML AUTO	•	•	•	•	•	•	•	
FOR XML PATH			•	•	•	•	•	•
FOR XML EXPLICIT	•				•	•	•	

Ukázka použití klauzule FOR XML:

```
select us.name as uzivatel, us.login, r.name as role
from User as us
inner join Role as r on
           us.role_id=r.id
for xml auto, XMLDATA
```

Výstup relačních dat viz obr. 3.2:

	uzivatel	login	role
1	Radka Bábíčková	babickova	učitel
2	Radka Bábíčková	xbabic02	student
3	Janko Hraško	xhrasko01	student
4	Jan Novák	xnovak03	student
5	Jana Holá	xhola02	student
6	Zuzana Nová	xnova02	student
7	Josef Nový	novy	učitel

Obrázek 3.2: Výstup relačních dat

Výstup dat ve formátu XML režim auto bez parametru:

```
<us uzivatel="Radka Bábíčková" login="babickova">
  <r role="učitel"/>
</us>
<us uzivatel="Radka Bábíčková" login="xbabic02">
  <r role="student"/>
</us>
<us uzivatel="Janko Hraško" login="xhrasko01">
  <r role="student"/>
</us>
```

```
<us uzivatel="Jan Novák" login="xnovak03">
  <r role="student"/>
</us>
<us uzivatel="Jana Holá" login="xhola02">
  <r role="student"/>
</us>
<us uzivatel="Zuzana Nová" login="xnova02">
  <r role="student"/>
</us>
<us uzivatel="Josef Nový" login="novy">
  <r role="učitel"/>
</us>
```

Výstup dat ve formátu XML režim auto s parametrem XMLDATA:

```
<Schema name="Schema2" xmlns="urn:schemas-microsoft-com:xml-data"
 xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="us" content="eltOnly" model="closed" order="many">
   <element type="r" maxOccurs="*"/>
    <AttributeType name="uzivatel" dt:type="string"/>
    <AttributeType name="login" dt:type="string"/>
    <attribute type="uzivatel"/><attribute type="login"/>
   </ElementType>
  <ElementType name="r" content="empty" model="closed">
    <AttributeType name="role" dt:type="string"/><attribute type="role"/>
  </ElementType>
</Schema>
<us xmlns="x-schema:#Schema2" uzivatel="Radka Bábíčková" login="babickova">
  <r role="učitel"/>
</us>
<us xmlns="x-schema:#Schema2" uzivatel="Radka Bábíčková" login="xbabic02">
  <r role="student"/>
</us>
<us xmlns="x-schema:#Schema2" uzivatel="Janko Hraško" login="xhrasko01">
  <r role="student"/>
</us>
<us xmlns="x-schema:#Schema2" uzivatel="Jan Novák" login="xnovak03">
  <r role="student"/>
</us>
<us xmlns="x-schema:#Schema2" uzivatel="Jana Holá" login="xhola02">
  <r role="student"/>
</us>
<us xmlns="x-schema:#Schema2" uzivatel="Zuzana Nová" login="xnova02">
  <r role="student"/>
</us>
<us xmlns="x-schema:#Schema2" uzivatel="Josef Nový" login="novy">
  <r role="učitel"/>
</us>
```

Výstup dat ve formátu XML režim raw bez parametru:

```
<row uzivatel="Radka Bábíčková" login="babickova" role="učitel"/>
<row uzivatel="Radka Bábíčková" login="xbabic02" role="student"/>
<row uzivatel="Janko Hraško" login="xhrasko01" role="student"/>
<row uzivatel="Jan Novák" login="xnovak03" role="student"/>
<row uzivatel="Jana Holá" login="xhola02" role="student"/>
<row uzivatel="Zuzana Nová" login="xnova02" role="student"/>
<row uzivatel="Josef Nový" login="novy" role="učitel"/>
```

Výstup dat ve formátu XML režim raw s parametrem ELEMENTS:

```
<row>
  <uzivatel>Radka Bábíčková</uzivatel>
  <login>babickova</login>
  <role>učitel</role>
</row>
<row>
  <uzivatel>Radka Bábíčková</uzivatel>
  <login>xbabic02</login>
  <role>student</role>
</row>
<row>
  <uzivatel>Janko Hraško</uzivatel>
  <login>xhrasko01</login>
  <role>student</role>
</row>
<row>
  <uzivatel>Jan Novák</uzivatel>
  <login>xnovak03</login>
  <role>student</role>
</row>
<row>
  <uzivatel>Jana Holá</uzivatel>
  <login>xhola02</login>
  <role>student</role>
</row>
<row>
  <uzivatel>Zuzana Nová</uzivatel>
  <login>xnova02</login>
  <role>student</role>
</row>
<row>
  <uzivatel>Josef Nový</uzivatel>
  <login>novy</login>
  <role>učitel</role>
</row>
```

3.1.2 Klauzule FOR XML WITH NAMESPACES

Rozšíření klauzule FOR XML o možnost přidávání jmenných prostorů. XML jmenné prostory se využívají k zabránění kolizím při pojmenování XML uzlů a k zlepšení flexibility

XML dokumentů.

Jmenné prostory jsou definované jejich URI (Uniform Resource Identifier) identifikátorem. URI jmenným prostorem může být jakýkoliv řetězec hodnot, ale většinou to bývá URL (Uniform Resource Locator) nebo webová adresa. URL se používá z důvodu jeho vysoké unikátnosti, což pomáhá zabránit konfliktům názvů jmenných prostorů, a mohou ukazovat na podpůrnou dokumentaci nebo definici XML schématu daného XML dokumentu. Prefix XML jmenného prostoru je pojmenování pro jmenný prostor, ale při zpracování je rozšířen na kompletní URI odpovídající danému jmennému prostoru.

Například máme prefix jmenného prostoru ns1 a URI http://www.microsoft.com/AdventureWorksDB/Person. XML procesor rozšíří jméno prvku ns1:Person na jméno: {http://www.microsoft.com/AdventureWorksDB/Person}Person, proto je nesmírně důležité, aby jmenný prostor měl unikátní URI, [3].

3.1.3 Klauzule OPENXML

Funkce OPENXML vkládá do tabulky data z XML dokumentu. Funkce OPENXML převede po analýze obsahu XML dokumentu, tento dokument na rowset. Takto zpracovan8 data ve výsledném rowsetu lze uložit do konkrétních databátových tabulek nebo pohledů.

Syntaxe OPENXML klauzule [16]:

```
OPENXML(idoc int [in] , rowpattern nvarchar [in] , [ flags byte [in] ])
[WITH (SchemaDeclaration | TableName)]
```

přičemž:

- **idoc** - manipulátor interní reprezentace dokumentu. Vnitřní reprezentace XML dokumentu je vytvořena voláním sp_xml_preparedocument.
- **rowpattern** - je výraz XPath sloužící k identifikaci uzlů ve zdrojovém XML dokumentu, které mají být zpracovány jako databázové řádky.
- **flags** - je volitelný vstupní parametr. Určuje jakým způsobem budou zpracovány atributy a elementy ve vybraných uzlech. Možné hodnoty jsou v tabulce 3.2.
- **SchemaDeclaration** - definice formy schématu cílové tabulky: ColName ColType [ColPattern — MetaProperty] [, ColNameColType [ColPattern — MetaProperty]...] popis v tabulce 3.3.
- **TableName** - název cílové tabulky. Pokud tabulka s definovaným schématem již existuje, stačí zadat pouze název tabulky.

Podpora OPENXML v SQL Serveru je zachována z důvodu zpětné kompatibility dříve napsaných kódů. V současné době byla tato funkčnost nahrazena metodami MS SQL Serveru 2008 pro datový typ XML. Tyto metody mají lepší kontrolu, flexibilitu a snadné používání díky silnějšímu XQuery jazyku pro práci s XML daty.

Tabulka 3.2: Hodnoty OPENXML flags parametru, převzato z [3]

Hodnota	**Popis**
0	Výchozí hodnota, pokud není v OPENXML nastaven parametr flags. Definuje mapování zaměřené na atributy.
1	Definuje mapování zaměřené na atributy (attribute-centric mapping).
2	Používá se v případě mapování zaměřeného na elementy (element-centric mapping).
3	Kombinace 1 a 2. Nejdříve se provede mapování zaměřené na atributy a pak se použije na dosud neřešené sloupce mapování zaměřené na elementy.
8	Může být kombinováno (logické OR) s XML_ATTRIBUTES nebo XML_ELEMENTS.
9	Tento parametr kombinuje 1 a 8 pro provádění mapování zaměřeného na atributy a udává, že zpracovaná data by neměla být kopírována, aby nedošlo k přetečení.
10	Kombinace 2 a 8 pro provádění mapování zaměřeného na elementy a udává, že zpracovaná data by neměla být kopírována, aby nedošlo k přetečení.
11	Kombinuje 1, 2 a 8. Jako první provede mapování zaměřené na atributy a následně se použije na dosud neřešené sloupce mapování zaměřené na elementy. Z toho také vyplývá, že zpracované údaje by neměly být kopírovány z důvodu přetečení.

Tabulka 3.3: Formy schématu cílové tabulky:

Forma	**Popis**
ColName	Název sloupce v rowset.
ColType	Datový typ sloupce v rowset. Pokud je typ sloupce jiný jak základní XML datový typ atributu, pak je typ násilně ukončen.
ColPattern	Volitelný parametr. Obecný XPath model, který popisuje, jak by se měly XML uzly mapovat do sloupců. Pokud není tento parametr specifikován, tak probíhá mapování, které je definováno prostřednictvím parametru flags. XPath model definuje speciální možnosti mapování, které nahrazují nebo rozšiřují mapování nastavené prostřednictvím parametru flags.
MetaProperty	Jedna z metavlastností, které umožňuje funkce OPENXML. Pokud jsou specifikovány, pak sloupce obsahují informace poskytnuté metaproperty. Prostřednictvím meta vlastností lze získat další informace o XML uzlech, jako jsou například relativní pozice a informace o jmenném prostoru. Toto poskytuje více infomrací, než je obsaženo v textové podobě.

Ukázka použití klauzule OPENXML:

```
declare @idoc int;

declare @xmldocument varchar(1000)
set @xmldocument = '
<users>
  <us name="uzivatel1" login="uzivatel1" password="1234" role_id="1">
  </us>
  <us name="uzivatel2" login="uzivatel2" password="4321" role_id="1">
  </us>
  <us name="uzivatel3" login="uzivatel3" password="5678" role_id="2">
  </us>
  <us name="uzivatel4" login="uzivatel4" password="8765" role_id="2">
  </us>
</users>';

EXECUTE sp_xml_preparedocument @idoc OUTPUT, @xmldocument;

SELECT    *
FROM  OPENXML (@idoc, '/users/us',1)
              WITH (name nvarchar(255),
                    login nvarchar(50),
                    password nvarchar(50),
                    role_id int)
```

Výstup relačních dat z XML viz obr. 3.3:

	name	login	password	role_id
1	uzivatel1	uzivatel1	1234	1
2	uzivatel2	uzivatel2	4321	1
3	uzivatel3	uzivatel3	5678	2
4	uzivatel4	uzivatel4	8765	2

Obrázek 3.3: Výstup relačních dat z XML

3.2 Datový typ XML

Datový typ XML byl poprvé skutečně integrován v MS SQL Serveru 2005 jako první datový typ zkonstruovaný pro manipulaci, ukládání a dotazování XML dat. XML datový typ poskytuje mnoho výhod oproti stávajícím funkcím a poskytovaným procedurám pro práci s XML na straně serveru ve starších verzích MS SQL Serveru. XML datový typ je typem LOB (Large Object) a každá XML instance může mít velikost až 2,1 GB.

XML datový typ lze použít jako XML proměnnou, typ sloupce v tabulce, typ parametru v uložené proceduře a nebo jako typ návratové hodnoty v uživatelsky definovaných funkcích (user-defined function).

XML datový typ může být naplněn jinými datovými typy pomocí T-SQL (Transact-SQL) a jeho funkcí CAST a CONVERT. Funkce CONVERT plní stejnou funkci jako CAST, ale CONVERT nabízí další možnosti, jako použití DTD (Document Type Definition) a zachování nevýznamných mezer (white-space).

Následující data mohou být naplněna do XML datového typu:

- **datový typ varchar** - může být převeden implicitně i explicitně na XML datový typ. Není vyžadována deklarace kódování XML. Pro 8-bitové kódování je výchozí utf-8. Jinak lze použít pro 8-bitové kódování iso-8859-1 nebo windows-1252.
- **datový typ nvarchar** - lze ho převést implicitně či explicitně na XML datový typ. Nelze použít 8-bitové kódování XML, ale lze použít 16-bitové kódování. Jako výchozí kódování se aplilkuje utf-16.
- **datový typ varbinary** - taktéž může být převeden implicitně i explicitně na XML datový typ. Když varbinary data neobsahují pořadí byte znaků (byte order mark), tak se použije 8-bitové kódování a naopak, pokud obsahuje pořadí byte znaků, tak se apllikuje 16-bitové kódování. Při použití 16-bitových dat s 8-bitovým kódováním specifikovaném v deklaraci XML může vést k podivným a neočekávaným výsledkům obsahujícím mezinárodní znaky. Tato situace může nastat, když se převedou nvarchar data na varbinary data a tyto se ještě převedou na XML datový typ.
- **datové typy char, nchar a binary** - mohou být rovněž převedeny na XML datový typ implicitně či explicitně. Pokud se použijou binární data, tak XML data musí obsahovat přesnou definici délky dat. Proměnná délka datového typu má tendenci být mnohem pružnější než jejich protějšek s pevnou délkou dat.

XML data mohou být reprezentována dobře strukturovanými (well-formed) dokumenty, a to buď s definovaným (Typed) nebo nedefinovaným typem (Untyped), nebo XML fragmenty. XML fragment je XML instance, které chybí uzel nejvyšší úrovně, čili kořenový uzel.

Server definuje dvě charakteristiky pro XML datový typ DOCUMENT pro well-formed XML dokumenty a CONTENT pro XML fragmenty. Když jsou XML sloupce, parametry a proměnné deklarované bez charaktesristiky, tak se jako výchozí charakteristika nastaví CONTENT. Charakteristika je v tomto případě chápána jako omezení nebo zákaz v XML obsahu.

Untyped XML je jednoduché XML, které nemá přiřazené XML schéma. Typed XML data mohou být uložena ve sloupcích s definovaným XML datovým typem a mají definovaná XML schémata, více o XML schématech viz kapitola 3.3.

Kromě XML schématu podporuje MS SQL Server 2008 i malou množinu z XML DTD (Document Type Definition). DTD se definuje na začátku dokumentu hned za XML deklarací. Podpora DTD v SQL Serveru je omezena na dvě funkce:

1. přiřazení výchozí hodnoty atributům v XML datech.
2. rozšíření referencí entit v XML datech

Standard vytvořený konsorciem W3C definuje další funkce DTD, včetně použití DTD pro definování struktury a textového obsahu elementů a atributů v XML datech. MS SQL Server tyto funkce nepodporuje, ale lze pro tyto účely využít kolekci XML schémat.

3.2.1 Metody XML datového typu

Metody související s XML datovým typem pomáhají v inetgraci XML do relačního prostředí MS SQL Serveru. Metody a jejich popis je v následující tabulce 3.4 [3], [16]:

Tabulka 3.4: Metody XML datového typu

Metoda	Popis
query()	Umožňuje provést XQuery nad instancí XML datového typu. Výsledkem je XML typ. Metoda vrací instanci untyped XML.
value()	Provádí XQuery nad instancí XML a vrací skalární hodnotu. Obvykle se využívá pro výpis hodnot z XML instance, která je uložena ve sloupci, parametru nebo proměnné XML datového typu. Tímto způsobem lze zadat SELECT dotazy, které kombinují nebo porovnávají XML data s daty ve sloupcích, které neobsahují XML data, čili nejsou typu XML.
exist()	Dotaz nad XML daty a vrací bitovou hodnotu. Když vrátí hodnotu 1, čili true, což znamená , že XQuery výraz v dotazu vrátí neprázdný výsledek. Neprázdný výsledek znamená, že je nalezen alespoň jeden XML uzel. Hodnota 0, reprezentuje false hodnotu, čili nebyl nalezen žádný XML uzel. Byl vrácen prázdný výsledek. Výsledekem ještě může být i hodnota NULL v případě když i XML instance obsahuje hodnotu NULL.
modify()	Používá se pro změnu obsahu proměnných a sloupců, které jsou XML datového typu. Tato změna se provádí pomocí XML DML (XML Data Manipulation Language). Tato metoda se musí používat se SET klauzulí nebo výrazem.
nodes()	Využívá se , když je zapotřebí převést XML data do relační podoby. Výsledkem metody je rowset, který obsahuje logickou kopii originálních instancí XML. V těchto kopiích uzel souvisí s každým řádkem instance a je nastaven na jeden z uzlů identifikujících výraz dotazu, takže následující dotazy se mohou vzájemně navigovat k jejich kontextovým uzlům.

Jak je vidět metody XML datového typu spoléhají na SQL Server XQuery a funkčnost XML DML. XQuery je zase závislé na implementaci W3C XDM (XQuery and XPath Data Model) v SQL Serveru.

3.3 XML schémata

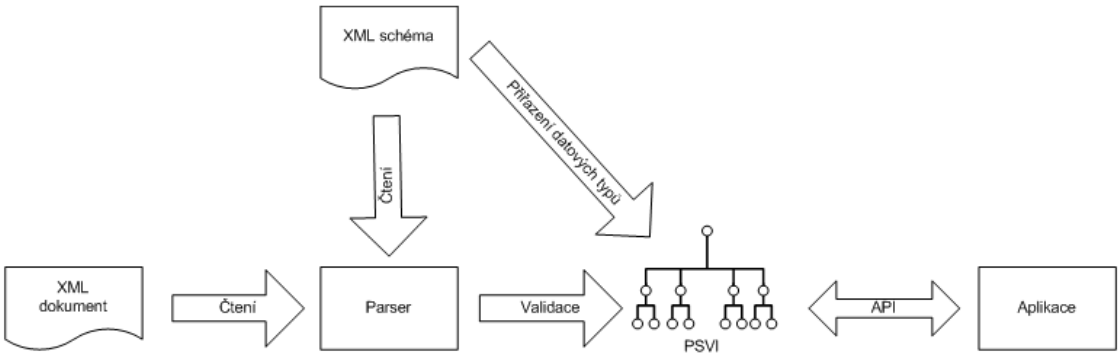
XML schéma je norma W3C konsorcia, která stanovuje formální strukturu a omezení obsahu XML dat. Výhodou této definice je jednoznačnost, která znemožňuje různé interpretace. Jelikož schéma jednoznačně definuje jak může dokument XML vypadat, z tohoto důvodu ji lze využít i pro validaci XML dokumentu. Tímto způsobem si můžeme ověřit správnost formátu XML dokumentu, který nám například zaslal obchodní partner a na kterém jsme se pro vzájemnou komunikaci dohodli. XML schémata jsou mnohem pružnějším a výkonnějším způsobem validace XML dokumentů než starší metoda využívající DTD (Document Type Definition).

SQL Server podporuje CREATE XML SCHEMA COLLECTION, DROP XML SCHEMA COLLECTION, a ALTER XML SCHEMA COLLECTION klauzule pro registraci a správu XML

schémat uložených v databázi. XML schémata jsou definovány prostřednictvím formálního XSD (XML Schema Definition) jazyka, který obsahuje prostředky pro definování struktury a různých omezení obsahu XML dat. XSD je definován prostřednictvím standardní syntaxe XML, tudíž XML schéma dokument je vlastně jen jiný druh XML dokumentu.
XSD dokument nastavuje následující vlastnosti XML dokumentu:

- pořadí elementů a atributů
- hierarchickou strukturu elementů v XML dokumentu
- určuje zda daný element v dokumentu může obsahovat text
- definuje datový typ elementů a atributů v dokumentu
- vymezuje výchozí a pevně dané hodnoty elementů a atributů v dokumentu

Po nastavení datového typu jednotlivým částem XML dokumentu, se pak pracuje rovnou s otypovanými daty a ne jen s textovými řetězci. Z abstraktního datového modelu XML tzv. Infoset se takto stane Post-Schema Validation Infoset (PSVI). Nad PSVI pracují dotazovací jazyky, jako je XQuery nebo XPath. Na obrázku 3.4 je zobrazen vznik PSVI.



Obrázek 3.4: Parser zpřístupňující PSVI, převzatý z [5]

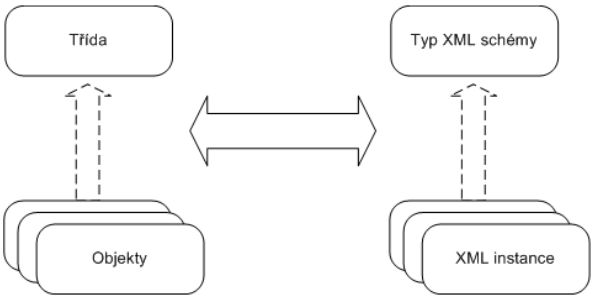
XML schéma podporuje dvě základní skupiny datových typů:

1. jednoduché typy - definování jednoduchého obsahu elementu nebo hodnoty atributu. Patří mezi ně primitivní typy, které představují datové typy, které nemohou být odvozeny z jiných datových typů. Jedná se o základní datové typy, jako jsou boolean, float nebo decimal.
2. komplexní typy - definují strukturu elementu, včetně platných atributů, obsahu a synovských uzlů elementu, čili můžou se skládat z alemetů a atributů. Patří mezi ně uživatelsky definované typy, které popisují elementy se synovskými elementy nebo atributy.

Propojení XML dokumentů a schémat:

1. Instance XML je oddělena od schématu, což lze přirovnat k objektově-orientovanému (OO) vztahu mezi třídami a objekty, viz obr. 3.5.

   Jedná se o koncepční posun od způsobu práce prostřednictvím DTD.



Obrázek 3.5: OO vs. XML koncept, převzatý z [10]

2. Napojení XML schématu prostřednictvím in-line kódu nebo externího odkazu, což zjednodušuje fázi návrhu a taktéž i transport dat, protože všechny informace jsou na jednom místě.

   (a) noNamespaceSchemaLocation - pokud se nepoužívají jmenné prostory, tak se do tohoto atributu uvádí URL adresa schématu. Lze použít i relativní URL a odkázat se na schéma, které je ve stejném adresáři jako dokument.

   ```
   <dokument
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="nazev.xsd">
     ...
   </dokument>
   ```

   (b) schemaLocation - používáme v případě jmenných prostorů. Lze zadat několik dvojic URI jmenného prostoru a umístění schématu, tyto hodnoty jsou oddělené mezerami nebo jinými bílými znaky.

   ```
   <dokument
     xmlns="URI"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="URI
                         nazev.xsd">
     ...
   </dokument>
   ```

Ukázka CREATE XML SCHEMA COLLECTION:

```
CREATE XML SCHEMA COLLECTION
QuestionSchema AS N'
<?xml version="1.0" encoding="UTF-16"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"  attributeFormDefault="unqualified">
 <xs:element name="question">
    <xs:annotation>
      <xs:documentation>otazka</xs:documentation>
    </xs:annotation>
    <xs:complexType>
```

```
      <xs:sequence>
        <xs:element name="id" type="xs:integer"/>
        <xs:element name="text" type="xs:string"/>
        <xs:element name="type">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="1zN"/>
              <xs:enumeration value="NzN"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>' ;
GO
```

3.4 XQuery

XQuery je dotazovací jazyk pro dotazování XML dat navržený a spravovaný konsorciem W3C. Do MS SQL Serveru 2008 byla zabudována podpora XQuery i prostřednictvím metod XML datového typu.

MS SQL Server XML tým zavedl pevnou množinu XQuery funkcí a operátorů doporučených W3C konsorciem. Součástí MS SQL Serveru je úplný soubor logických operátorů, většina matematických operátorů, základní množina standardních XQuery funkcí a některé specifické MS SQL Server funkce.

XQuery je postaveno na syntaxi některých dřívějších dotazovacích XML jazykú, a to Quilt, XPath 1.0, XQL, a XML-QL, včetně databázových dotazovacích jazyků SQL a OQL. Technicky vzato je XQuery považováno za rozšíření jazyka XPath, nebo taky XPath je podmnožinou jazyka XQuery. Tyto jazyky mají shodnou syntaxi a gramatiku a mají stejný datový model (XDM). Je zaručeno, že jakýkoliv platný XPath výraz vrátí ten samý výsledek jako XQuery.

XQuery je funkcionální jazyk, jehož základní jednotkou jsou výrazy. Jak již bylo zmíněno jazyk, XQuery je složen z více jazyků a z toho vyplývají i jeho vlastnosti, jako je podpora různých typů výrazů, které mohou být složeny z klíčových slov, symbolů a operandů. Výrazy lze libovolně mezi sebou kombinovat, zanořovat nebo využívat jejich výsledek jako parametr dalšího výrazu.

Jazyk XQuery zachovává uzavřenost nad datovým modelem, je case sensitive, má silnou typovou kontrolu, která vede k větší bezpečnosti a někdy i k složitějšímu psaní výrazů. XQuery je úzce provázáno s XML schématy prostřednictvím, kterých se definují datové typy.

Jiří Kosek definoval jazyk XQuery následovně [6]:
XQuery = XPath 2.0 + FLWOR výrazy + výrazy konstruující nové elementy + uživatelsky definované funkce + několik dalších direktiv

3.4.1 XDM

Základní datový model XDM definuje abstraktní model a typový systém, který používá XQuery pro provádění dotazů a operací nad XML daty. Specifikace XDM rozšiřuje možnosti datových typů v XML schématu datovými typy, jako jsou untyped, untypedAtomic, anyAtomicType, dayTimeDuration, and yearMonthDuration. Nicméně MS SQL Server nepodporuje dayTimeDuration a yearMonthDuration.

XDM mapuje XML obsah do hierarchické struktury a podporuje sedm druhů uzlů pro mapování:

- **dokumentové uzly** - zapouzdřují XML dokumenty. Synovské uzly mohou obsahovat elementové uzly, uzly zpracovávající instrukce, komentářové a textové uzly. Tyto uzly nemají rodičovský uzel, čili je lze přirovnat k samotnému rodičovskému uzlu. Rozdíl oproti XQuery je v tom, že XQuery lze použít na fragmenty XML. XML data, která nejsou dobře strukturovány, mohou mít více dokumentových uzlů.
- **elementové uzly** - reprezentují XML elementy. Mohou mít rodičovský uzel. Synovské uzly obsahují elementové uzly, uzly zpracovávající instrukce, komentářové a textové uzly, tak jako uzly dokumentové.
- **atributové uzly** - představují XML atributy. Atributové uzly mají odpovídající elementový uzel, i když ho vždy mít nemusí, tak jako ani rodičovský uzel.
- **uzly jmenných prostorů** - odpovídají URI jmenného prostoru, odkazujícího na prefix jmenného prostoru nebo výchozí jmenný prostor.
- **uzly zpracovávající instrukce** - reprezentují XML zpracovávající instrukce.
- **komentářové uzly** - představují XML komentáře.
- **textové uzly** - odpovídají obsahu XML textu.

MS SQL Server neukládá přesnou reprezentaci XML dat, ale místo toho převádí literál XML dat do vnitřní podoby formátu XDM. XDM poskytuje hierarchickou strukturu, která ukládá pouze důležité informace o XML uzlech. Některá z metadat se bohužel ztrácí v procesu konverze. Ztracené položky obsahující CDATA specifikací, znaky a DTD referenci, jsou doplněny v průběhu zpracování. Položky s číselnou hodnotou mohou být uloženy ve formátu nonliteral.

3.4.2 FLWOR výrazy

XPath patří mezi docela silné dotazovací jazyky, ale zápis složitějších podmínek může být nepřehledný, nelze seřadit výsledek výrazu a nejde generovat nové elementy na výstupu dotazu. Všechny tyto nevýhody odstraňují výrazy FLOWR (For-Let-Where-Order by-Return).

Struktura FLWOR výrazu:

- **For** - výběr posloupnosti uzlů ke zpracování, přiřazení proměnných
- **Let** - přiřazení proměnné výsledek k výrazu
- **Where** - definování omezujících podmínek výběru

- **Order by** - setřídí výsledek dle daného kritéria
- **Return** - specifikace výstupu pro každý vybraný uzel

3.4.3 XML DML

XML DML (XML Data Modification Language) je rozšíření jazyka XQuery o funkce pro manipulaci s XML daty, jako je aktualizace, mazání a vkládání uzlů. XML DML rozšiřuje XQuery o nasledující funkce:

- **insert** - vkládá jeden nebo více uzlů do již existující XML instance. Je obdobou SQL INSERT klauzule [16]. Syntaxe:

  ```
  insert výraz1 (  {as first | as last} into | after | before výraz2 )
  ```

  výraz1 - jeden nebo více uzlů ke vložení.
  {*as first* — *as last*} *into* — *after* — *before* - umístění uzlu, kam se má uzel vložit.
  výraz2 - označuje uzel, ke kterému má být uzel z výrazu1 vložen.

- **replace value of** - nahrazuje hodnotu výrazu. Je obdobou SQL UPDATE klauzule. Syntaxe:

  ```
  replace value of výraz1 with výraz2
  ```

  výraz1 - vybraný uzel pro modifikaci. Udává se pouze jeden uzel.
  výraz2 - nová hodnota uzlu.

- **delete** - odstraňuje uzel z XML instance. Je obdobou SQL DELETE klauzule. Syntaxe:

  ```
  delete výraz
  ```

  výraz - označuje uzly, které mají být smazány. Jsou smazány všechny uzly, a také všechny uzly a hodnoty, které jsou obsaženy ve vybraných uzlech. Nemůže být použit na hlavní (root) uzel.

3.5 Indexace

Indexace se využívá pro vytváření indexů nad daty databáze, tyto indexy pak zrychlují vyhledávání v datech. Běžné způsoby indexování relačních dat (B-tree) nejsou vhodné pro indexování XML dat, a proto byly vyvinuty speciální indexovací mechanizmy, použitelné pouze pro XML data. XML indexy mohou být vytvářeny nad sloupci s XML datovým typem. Lze indexovat všechny tagy, hodnoty a cesty (paths) nad XML instancemi ve sloupcích, což zlepšuje dotazování. XML indexy lze rozdělit na primární, sekundární a full-text indexy.

3.5.1 Primární XML index

Pro vytvoření primárního indexu musí mít tabulka, ve které se nacházejí XML sloupce, sdružený primární klíč. Toto zajistí, že pokud je tabulka segmentovaná, tak je stejně segmentovaný i primární index. MS SQL Server používá primární klíč k propojení řádků primárního indexu s řádky v tabulce s XML sloupci. Další omezení primárního indexu:

- Nelze vytvořit primární XML index nad výpočetním sloupcem, XML proměnnou nebo proměnnou pohledu nebo tabulky.
- Nelze vytvořit primární XML index nad sloupci, které nejsou typu XML.
- Může být vytvořen pouze jeden primární XML index na daném XML sloupci.
- Tabulka sice může obsahovat více primárních XML indexů, ale primární index obsahuje pouze jeden XML sloupec.
- Pokud se mění typ XML sloupce z untyped na typed, nebo naopak, tak je zaopotřebí nejdříve odstranit existující primární XML index.
- Jméno primárního indexu musí být unikátní a nesmí být stejné jako jméno relačního indexu v tabulce.
- Pro tvorbu primárního XML indexu musí být NSI_PADDING, ANSI_NULLS, QUOTED_IDENTIFIER, CONCAT_NULL_YIELDS_NULL, ANSI_WARNINGS, a ARITHABORT nastaveny na ON a NUMERIC_ROUNDABOUT nastaven na OFF.

Pro vytvoření primárního indexu se používá klauzule CREATE PRIMARY XML INDEX a ALTER INDEX a DROP INDEX klauzule umožňují správu stávajících primárních indexů.

Primární index se používá hlavně z důvodu opakovaných XQuery dotazů nad XML daty jakékoliv velikosti, velkými XML dokumenty nebo řádky obsahující velká XML data. Dále se využívá při tvorbě sekundárního indexu, který optimalizuje provádění specifických XQuery dotazů.

Primární index je členitá a stálá reprezentace XML BLOB (binary large object) v XML sloupcích. Pro každý XML BLOB objekt ve sloupci, index vytváří několik řádků dat. Počet řádků indexu je zhruba stejný s počtem uzlů v XML BLOB objektu. Když dotaz vrátí úplnou XML instanci, tak MS SQL Server poskytne instanci z XML sloupců. Dotazy používají v XML instancích primární XML index a mohou vrátit s použitím samotného indexu skalární hodnotu nebo XML podstrom.

Každý řádek obsahuje následující informace:

- Jméno Tagu - jméno elmentu nebo atributu.
- Hodnotu uzlu.
- Typ uzlu - elementový uzel, atributový uzel, textový uzel.
- Informace o dokumentu - reprezentované interním identifikátorem uzlu.
- Cestu od každého uzlu k hlavnímu (root) uzlu XML stromu. Tento sloupec je prohledáván path výrazem v dotazu.
- Primární klíč základní tabulky. Tento primární klíč je duplikován v primárním XML indexu pro zpětné spojení se základní tabulkou a maximální počet sloupců primárního klíče je omezen na 15.

3.5.2 Sekundární XML index

Zatímco primární index XQuery dotazu zlepšuje výkonnost eliminováním runtime odstraňování (shredding) XML dat, výkon může být pořád vylepšován v mnoha případech přidáním dalšího indexu. Tuto funkci poskytuje sekundární XML index.

Pro vytváření sekundárního indexu se používá klauzule CREATE XML INDEX a ALTER INDEX a DROP INDEX klauzule slouží pro správu stávajících sekundárních indexů.

Pro sekundární XML indexy jsou taktéž definovány různé omezení:

- Sekundární XML index lze vytvořit pouze nad sloupcem s existujícím primárním XML indexem.
- Omezení, které platí pro primární klíč, taktéž platí i pro sekundární XML klíč. Toto plyne z předchozí podmínky.
- Nelze smazat primární index, pokud nad ním existuje sekundární index. Nejdříve se musí odstranit sekundární index a až pak se může odstranit primární index

Sekundární XML indexy lze rozdělit do následujících skupin:

1. **PATH sekundární XML index** - je navržen k optimalizaci dotazů XQuery path. Primární index zlepšuje dotazování v případě použití exist() metody v části WHERE. Při použití PATH sekundárního indexu lze výkonnost těchto dotazů zlepšit. Primární index je XML index relační reprezentace (sloupců a řádků) XML sloupců. Zatímco PATH sekundární index vytváří nesdružený (nonclustered) index nad primárním indexem sloupců typu HID, VALUE, PK (Primary Key) a ORDPATH.
2. **VALUE sekundární XML index** - optimalizuje dotazy, kde je známá hodnota, ale jméno nebo umístění uzlu obsahující hodnoty nejsou známy v čase dotazování. Je postaven nad primárním indexem, který je vytvořen nad sloupci typu node value a path.
3. **PROPERTY sekundární XML index** - používá se při dotazech, které vrací jednu nebo více hodnot z jednotlivých XML instancí. Toto nastane, pokud se používá pro získání vlastností objektu metoda XML typu value(), a když je známa hodnota primárního klíče daného objektu. PROPERTY sekundární index je založen na primárních indexech, které jsou vytvořeny nad sloupci typu PK, path a node value.

3.5.3 Full-text XML index

Podpora Full-text indexu na straně MS SQL Serveru je již od verze 7.0, ale rozšíření na XML data přišlo až ve verzi MS SQL Serveru 2005. Full-text index lze vytvořit pomocí klauzule CREATE FULLTEXT INDEX. Zatímco primární a sekundární index indexují hodnoty a uzly, tak Full-text index indexuje čistě XML instance a ignoruje hodnoty, uzly a další XML syntaxi. Je povolen pouze jeden Full-text index pro tabulku, ale ne pro sloupec, protože Full-text index se používá na sloupce a ne tabulky.

Pro vytvoření Full-text indexu je zapotřebí, aby existoval primární index s unikárním primárním klíčem v tabulce, v které se bude používat Full-text XML index. Prvním krokem pro Full-text indexování XML sloupce je vytvoření Full-text katalogu pomocí klauzule CREATE FULLTEXT CATALOG, protože všechny Full-text indexy jsou v databázi uchovávané právě v tomto katalogu. Databáze může obsahovat jeden nebo i více Full-text katalogů.

Jakmile je úspěšně vytvořen Full-text index v XML sloupci, mohou se použít T-SQL CONTAINS a FREETEXT predikáty a CONTAINSTABLE a FREETEXTTABLE funkce pro provádění Full-textové vyhledávání v XML datech.

Full-text indexování umožňuje flexibilitu při hledání XML dat, protože ho lze použít na standardní word-breakers, stemmers a tezaurus funkce v XML obsahu v různých jazycích. Word-brakers aplikuje speciální pravidla jazyka na rozdělení Full-text indexovaného obsahu na tokeny o velikosti word. Většinou je dělícím znakem mezera. Stemmers představuje Full-text indexovací mechanizmus, který umožňuje vyhledávat varianty slov. Tezaurus funkce umožňuje vytvářet synonyma a podporuje záměnu a rozšiřování slov. Například se vytvoří tezaurus funkce, která slovem "červený"bude nahrazovat slova "rudý"a "karmínový".

3.6 SQLXML

SQLXML bylo navrženo jako propojovací technologie mezi XML a relačními daty, čili představuje propojení jazyka SQL a podpory pro práci s XML daty. Je to jazyk vyvýjený firmou Microsoft a není součástí standardu, čili neplést si jej s SQL/XML. SQLXML je k dispozici od verze MS SQL Serveru 2000. Mnoho funkcí SQLXML je v současnosti nahrazeno XML datovým typem a T-SQL a .NET funkcemi. SQLXML umožňuje vyvojářům konverzi relačních dat do XML formátu, dotazování a aktualizaci relačních dat prostřednictvím XPath a XML, a hromadné načítání XML dat do relačních tabulek. SQLXML byl původně navržen pro práci v neudržovaném kódu prostřednictvím OLEDB (Object Linking and Embedding Database), nicméně .NET Framework poskytuje řízené wrapper třídy pro SQLXML.
Možnosti SQLXML:

- **dotazování** - podporuje možnost provádět dotazy a výsledek dotazu nastavit na XML formát a to vše na straně klienta. Tato podpora na straně klienta je vhodná pro snížení zatížení na straně serveru. SQLXML dotazy se provádí pomocí Microsoft.Data.
  SqlXml.SqlXmlCommand objektu. SqlXmlCommand objekt poskytuje vlastnosti a metody přístupu .NET SQLXML funkcemi. Jelikož je tato třída wrapperem pro SQLXML funkce založená na OLEDB/COM, tak pro přpojení vyžaduje požívání OLEDB správce.

- **aktualizace - updategrams** - updategrams byl původně navržen pro vzdálenou manipulaci s daty, včetně aktualizace, vkládání a mazání dat prostřednictvím internetu. Updategrams je založen na mapování schématu s poznámkami. Jedná se o XML Data-Reduced (XDR) nebo XML Schema Definition (XSD) dokumenty s poznámkami a dalšími atributy, které definují mapování XML na relační data z updategrams.

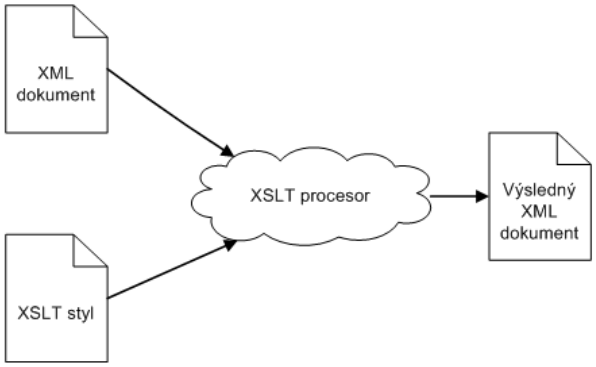
3.7 XSLT

XSLT je mocný a pružný jazyk určený k jednoduché manipulaci s XML dokumenty a jejich transformaci. XSLT je spravovaný konsorciem W3C a provádí transformaci XML dokumentů na jiný XML dokument.

XSLT umožňuje ve vstupním XML dokumentu přidávat nebo odebírat atributy, přeskupit a setřídit elementy, manipulovat s obsahem a rozhodnout o zpracování na základě testování uzlů a ostatních logických struktur. Jednou z nejvíce používaných transformací je XSLT transformace XML dokumentu do XHTML pro prezentaci na klientské vrstvě. Další

hojně používanou XSLT transfomrací je zpracování XML dat z různých zdrojů a vytvoření společného formátu.

Během XSL transformace jsou vstupní XML dokument a vybraný XSLT styl předány XSLT procesoru. XSLT styl je předpis podle, kterého budou data zpracovány. V XSLT procesoru se použije XSLT styl pro zpracování obsahu XML dokumentu a vytvoří výsledný XML dokument. Původní XML dokument zůstane nezměněn, protože veškeré změny jsou zaznamenány ve výsledném XML dokumentu. Proces XSL transformace je znázorněn na obr. 3.6.



Obrázek 3.6: Průběh XSL transformace, převzatý z [3]

XSLT používá XPath k vyhledávání, navigaci a k provádění testů uzlů ve zdrojovém dokumentu. XPath je taktéž pro XQuery path výrazy. XSLT je funkcionální jazyk, kdežto XSLT styl je XML dokument, který je složen z šablon. XSLT styl šablona je postavena z předdefinovaných XSLT prvků.

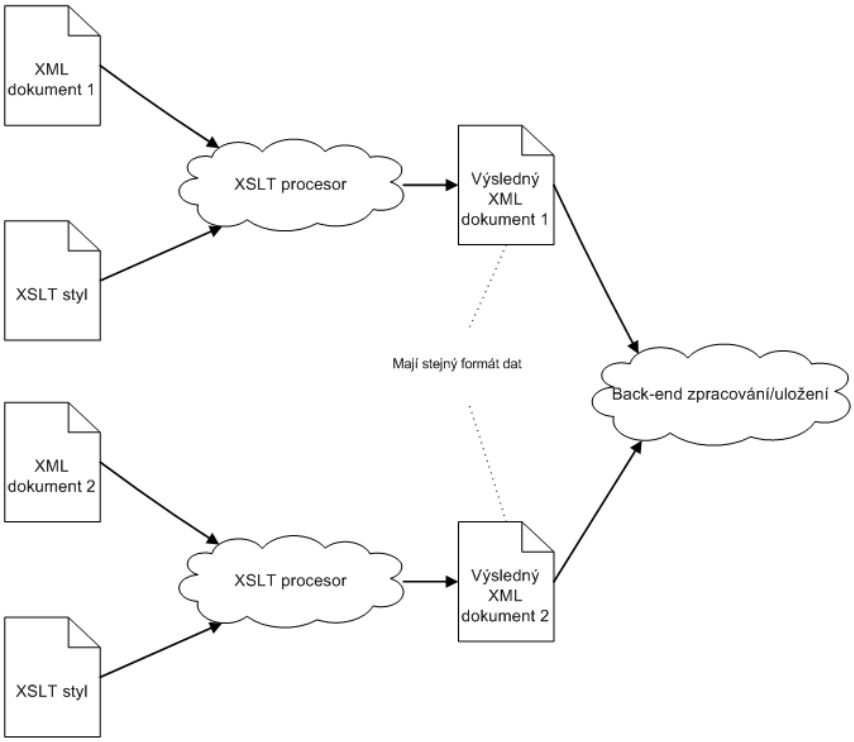
Na následujícím obrázku 3.7 je znázorněna tzv "back-end"XSL transformace. Jedná se o druhý nejčastější případ XSL transformace, a to jak již bylo zmíněno transformace XML dokumentů z různých zdrojů na stejný XML formát. Lze si představit případ kdy naše výrobky budou prodávat různí obchodní partneři a objednávky nám budou od nich chodit v různém formátu. Těžko je přesvědčíme, aby se přizpůsobili našemu formátu objednávky a tím změnili i svůj informační systém.

Jazyk T-SQL není postaven pro podporu XSLT, ale .NET framework tento nedostatek výborně zastoupí prostřednictvím svých System.Xml.Xsl jmenných prostorů. K této funkčnosti .NET frameworku lze snadno přistupovat pomocí MS SQL Serveru, ve kterém je integrován SQLCLR (SQL Common Language Runtime). Toto vyžaduje vytvoření a registraci .NET sady a vytvoření T-SQL funkcí a procedur pro přístup k funkčnostem této sady.

3.8 HTTP/SOAP Endpoints

MS SQL Server poskytuje podporu pro XML založeném na SOAP (Simple Object Access Protocol) endpoints přes HTTP. SOAP poskytuje schopnost serializovat objekty používající XML a HTTP je komunikačním protokolem, který odesílá a přijímá SOAP objekty. Jinak řečeno, instance MS SQL Serveru se může přímo použít jako webová služba serveru.

MS SQL Server poskytuje podporu pro vystavení uložených procedur a uživatelsky definovaných funkcí jako metody webových služeb používající SOAP. Používáním HTTP/SOAP endpoints lze mít vzdálený přístup k těmto procedurám a funkčnostem s minimálním úsilím.



Obrázek 3.7: Transformace různých XML dokumentů na stejný formát dat, převzatý z [3]

HTTP/SOAP endpoints byly zavedeny v MS SQL Serveru 2005, což se ukázalo jako významný pokrok od modelu MS SQL Serveru 2000. MS SQL Server 2000 spoléhal na IIS (Internet Information Services) a objekty založené na COM (Component Object Model) pro podporu webových služeb MSS SQL Serveru.

MS SQL Server umožňuje jednoduchou tvorbu a správu koncových bodů prostřednictvím příkazů a funkcí T-SQL, jako jsou CREATE ENDPOINT, ALTER ENDPOINT a DROP ENDPOINT.

Tato funkčnost bude v některé z dalších verzi MS SQL Serveru odstraněna, proto je už nyní vhodné využívat jiné protředky namísto HTTP/SOAP endpoints.

Visual Studio poskytuje jednoduché rozhraní pro přístup k metodám webových služeb a taktéž není obtížné přistupovat k webovým službám z jiných platforem a programovacích jazyků. Možnost přistupovat vzdáleně k funkcím z jakékoliv platformy a odkudkoliv, je opravdovou mocí webových služeb a SOA (Service Oriented Architecture).

Kapitola 4

Ukázková aplikace

Součástí práce je i praktická ukázka využití získaných vědomostí při studiu XML a jeho podpory na straně MS SQL Serveru 2008, s vhodně zvolenými technologiemi pro vývoj klientské části aplikace. Cílem praktické části diplomové práce je co nejvíce si vyzkoušet práci s XML u MS SQL Serveru, či už v rámci příkladů použitých v teoretické části diplomové práce nebo v rámci samotné ukázkové aplikace. Zadání ukázkové aplikace vznikalo až v průběhu studia MS SQL Serveru a jeho podpory XML. Z více návrhů byla vybrána Testovací aplikace pro tvorbu a správu testů součástí, které jsou samostatné části pro studenty a učitele.

4.1 Požadavky

4.1.1 Případy použití

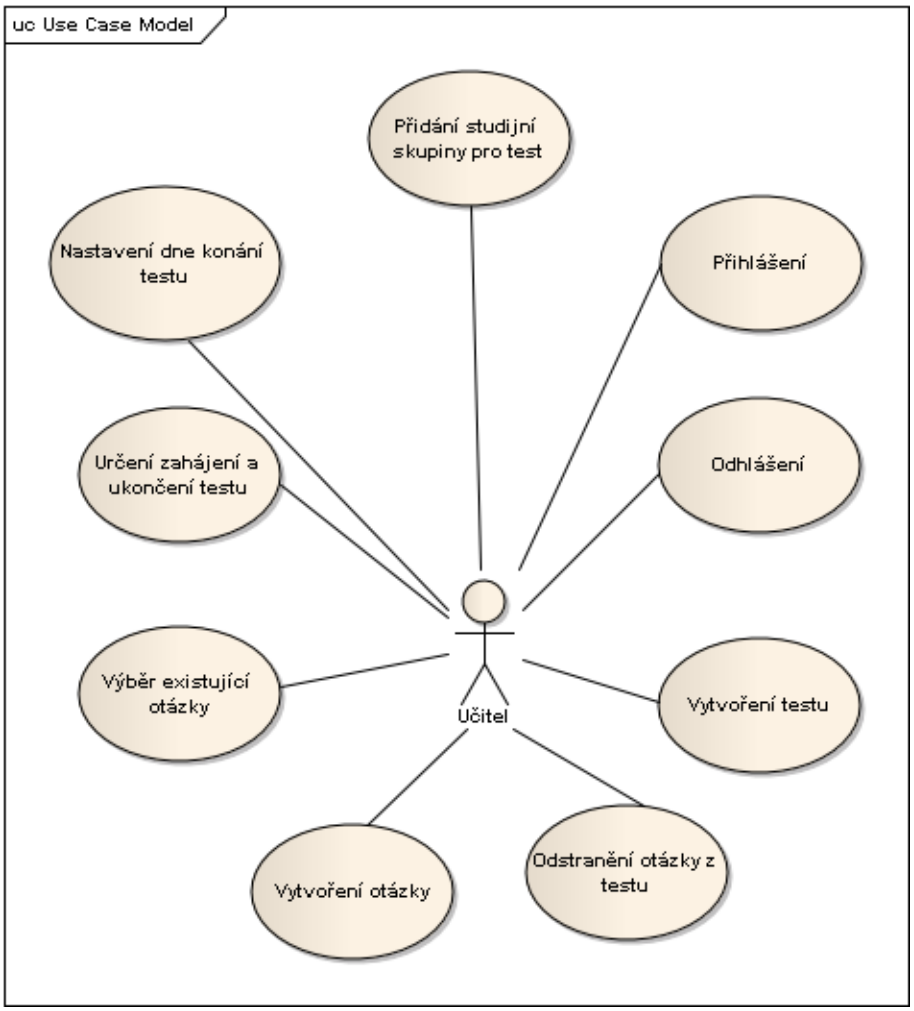
Možnosti využití aplikace jsou ilustrovány diagramy případů použití viz obr. 4.1 a obr. 4.2 a jejich popisem v této kapitole, kapitole [**?**] a kapitole [**?**].

Aplikace má dvě části, které odpovídají i samotným rolím uživatelů systému a to konkrétně část přístupná učitelům a část přístupná studentům. Každý uživatel má svůj přístupový účet do aplikace a určenou svou roli v systému. Uživatel se do aplikace přihlašuje svým loginem a heslem. Učitel má v aplikaci možnost vytvářet testy v rámci jednotlivých předmětů jejichž je garantem.

Test lze vytvářet pro konkrétní předmět (kategorii) a nelze jej zařadit do více kategorií najednou, protože každý předmět má své testy a tyto testy mají svůj okruh otázek. V systému lze vytvářet nové testy a přiřazovat jim otázky. Testy jsou složeny z jednotlivých otázek a jejich odpovědí. Otázky do testu lze vybírat z již existujících otázek v rámci dané kategorie a nebo lze do systému prostřednictvím importu přidat novou sadu otázek do dané kategorie.

Otázka může mít jednu nebo více správných odpovědí. Konkrétní odpovědi na otázku náleží pouze dané otázce a nelze je vybírat jako možnou odpověď na jinou otázku. Každý test má svoje parametry, jako je den konání testu a čas zahájení a ukončení testu.

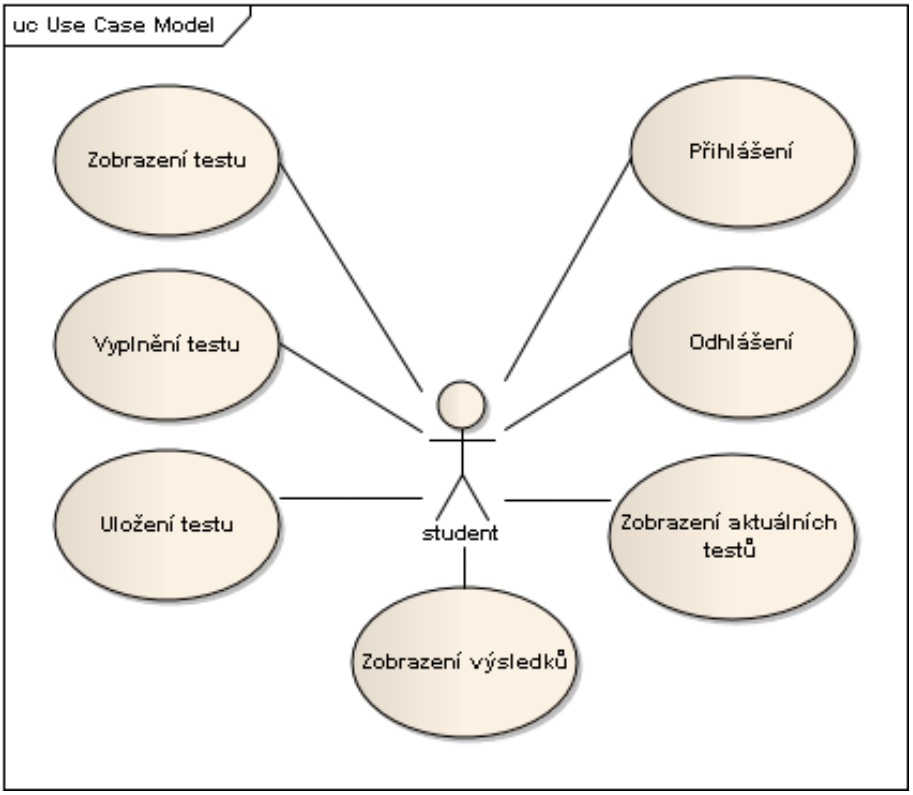
Testu lze přiřadit studijní skupinu, které bude daný test přístupný pro vyplnění.



Obrázek 4.1: Diagram případu použití - Učitelr

Studentovi se po přihlášení do aplikace nabídnou aktivní testy pro vyplnění. Jedná se o testy, které mají den konání stejný s aktuálním datem a taktéž kterým zrovna plyne čas konání, čili čas zahájení je menší než aktuální čas a zároveň aktuální čas ještě nedosáhl času ukončení testu.

Student si může zobrazít konkrétní test pro vyplnění odpovědí a následně vyplnit své odpovědi na otázky. Po vyplnění testu student své odpovědi uloží a test je vyhodnocen a výsledky testu studentovi zobrazeny.



Obrázek 4.2: Diagram případu použití - Student

4.1.2 Model domény

Model domény je znázorněn viz obr. 4.3. Jedná se o diagram tříd pro jednoduchost neobsahuje atributy a operace.

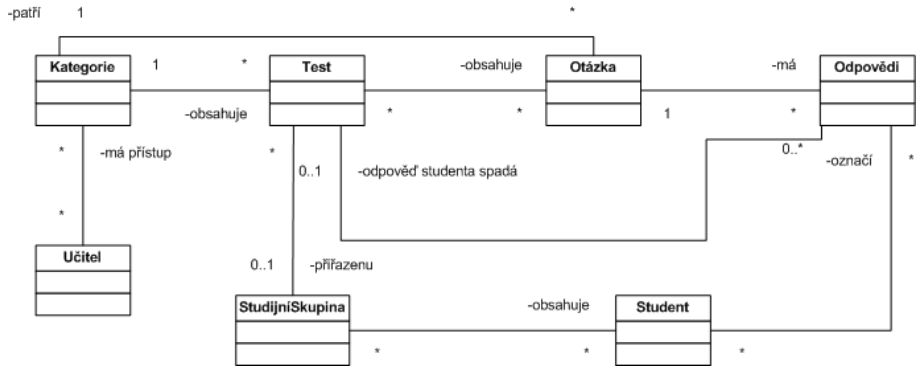
V aplikaci jsou dvě třídy uživatelů a to učitel a student. Učitel po přihlášení má přístup ke kategoriím, ke kterým má oprávnění a následně jsou mu po výběru kategorie zobrazeny testy, které tato kategorie obsahuje. Jak již bylo zmíněno na test jsou navázány jeho otázky a odpovědi, které se zobrazí po výběru konkrétního testu. Učitel si může taktéž zobrazit i parametry testu a studijní skupinu přiřazenou k testu. Učitel může s testem i pracovat tak, že do testu přidává nebo odebírá otázky, mění nastavení testu, případně odebírá nebo přidává studijní skupinu k testu.

Pro učitele je základní třídou kategorie, do které spadá test, a také samotný test a jeho součásti, učitel může importovat sadu otázek pro vybranou kategorii.

Student má po přihlášení přístup pouze k testům, ke kterým byl v rámci studijní skupiny přiřazen. Zobrazovat si obsah testu a odpovídat na testové otázky může pouze u aktivních testů, tzn že datum a čas vyměřený pro zodpovězení testu musí odpovídat aktuálnímu dni a aktuální čas musí spadat do nastaveného časového období. Student po zodpovězení otázek test uloží a ten je následně vyhodnocen a studentovi zobrazen jeho výsledek. Student pracuje přímo s testem, resp. s odpověďmi na jeho otázky, s testem je propojen přes třídu studijní skupina.

Vztahy mezi třídami mají násobnost many, až na vazbu mezi testem a kategorií, otázkou a kategorií a testem a studijní skupinou, protože test může spadat pouze do jedné kategorie

testů. Otázka taktéž patří pouze do jedné kategorie a test může mít přiřazenou pouze jednu studijní skupinu. Vazba mezi třídou otázka a kategorie je z důvodu přidávání a odebírání otázek testu. Pokud by tato vazba neexistovala, tak funkčnost výběru otázek z již existujících otázek by nebyla možná. Díky navržené vazbě lze z některých testů odebírat otázky, které se již nenachází v jiném testu dané kategorie a přitom je zachována možnost tyto otázky přidávat do jiných variant testů. Přímá vazba mezi odpovědí a testem existuje z důvodu přiřazení odpovědi studenta na danou otázku ke konkrétnímu testu. Tato vazba je zapotřebí z důvodu možnosti mít otázku s odpověďmi ve více testech a je zapotřebí přesně identifikovat, ke kterému testu se odpověď studenta vztahuje.



Obrázek 4.3: Diagram tříd

4.2 Návrh aplikace

4.2.1 Databáze

Základem celé aplikace je databáze, která využívá XML datový typ a podporu pro práci s XML. Schéma databáze vzniklo transformací modelu domény a obsahuje vztahy, atributy a integritní omezení viz obr. 4.4.

Struktura databáze je složena z následujících tabulek:

- *Role - číselník rolí uživatelů v sytému. V této aplikaci role učitel a student.*
- *User - tabulka s uživately sytému a jejich přihlašovacími údaji.*
- *StudyGroup - tabulka obsahující studijní skupiny, do kterých jsou zařazeni jednotliví studenti.*
- *User_StudyGroup - je vztahovou tabulkou pro vztah mezi uživatelem a studijní skupinou.*
- *Category - tabulka obsahuje všechny kategorie testů.*
- *Category_User - vztahová tabulka mezi uživatelem s rolí učitel a kategorií testů.*
- *Test - tabulka obsahující testy, resp. jejich základní nastavení.*
- *Test_StudyGroup - tabulka představující přiřazení studijních skupin k danému testu.*

- *Question - seznam otázek, které jsou obsaženy v různých testech a patří pouze do jední kategorie.*
- *Test_Question - tabulka obsahující záznamy o příslušnosti otázek k testům.*
- *Answer - včechny odpovědi, které jsou přiřazeny k jednotlivým otázkam.*
- *Answer_User - záznamy odpovědí jednotlivých studentů na otázky konkrétního testu.*

XML datový typ je využit pro uložení parametrů testu, otázek a odpovědí. Pro parametry testu je XML datový typ zvolen z důvodu optimalizace databáze, protože jinak by musely být parametry testu uloženy v samostatné tabulce. Jedná se o parametry den konání testu, začátek a konec testu. Další parametry testu lze získat z jiných tabulek a načítat je do XML až při případném exportu testu, nebo si je lze ukládat přímo do XML z důvodu optimalizace, jedná se o kategorii testu a studijní skupinu. Login studenta se do testu načte vždy až při exportu.

XML datový typ pro záznam otázek je zvolen z důvodu zachování formátu celého testu a XML obsahuje parametr id, který lze využít jako označení pořadí otázky v rámci testu, čímž se ušetří jeden sloupec v databázové tabulky.

U odpovědí na otázku obsahuje struktura XML odpovědi taktéž parametr id, který se striktně využívá pro záznam pořadí odpovědi v rámci otázky. Dále vybrané otázky, které představují správnou odpověď mají taktéž svůj parametr a tímto se ušetří další sloupec v databázové tabulce.

XML datový typ je zvolen i z důvodu jeho přenositelnosti mezi různými systémy.

4.2.2 Struktura XML

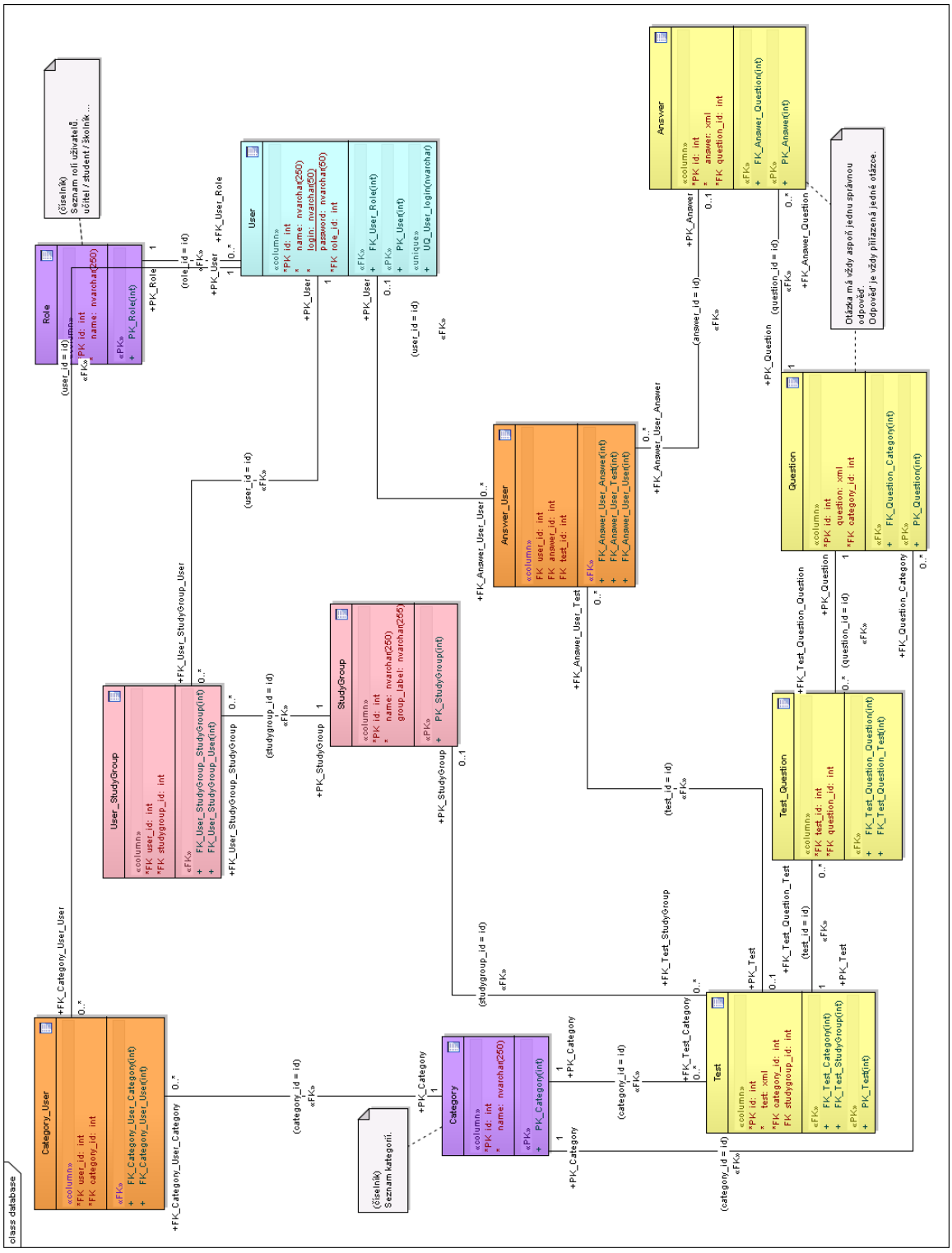
V dokumentu formátu XML jsou zohledněny včechny relevantní informace potřebné při exportu testu. Struktura viz obr. 4.5 a XML schéma viz příloha C.

Element *test* obsahuje elementy:

- *id* - identifikace testu, vnitřní označení testu, v případě, když je název testu stejný pro více testů.
- *name* - název testu.
- *category* - kategorie testu.
- *date* - den konání testu.
- *from* - čas začátku testu.
- *to* - čas ukonční testu.
- *group_name* - název studijní skupiny, v případě exportu zodpovězeného testu studentem.
- *login* - jméno/login studenta, v případě exportu zodpovězeného testu studentem.

Element *question* obsahuje elementy:

- *id* - identifikátor otázky, lze využít jako označení pořadí otázky.
- *text* - znění otázky.



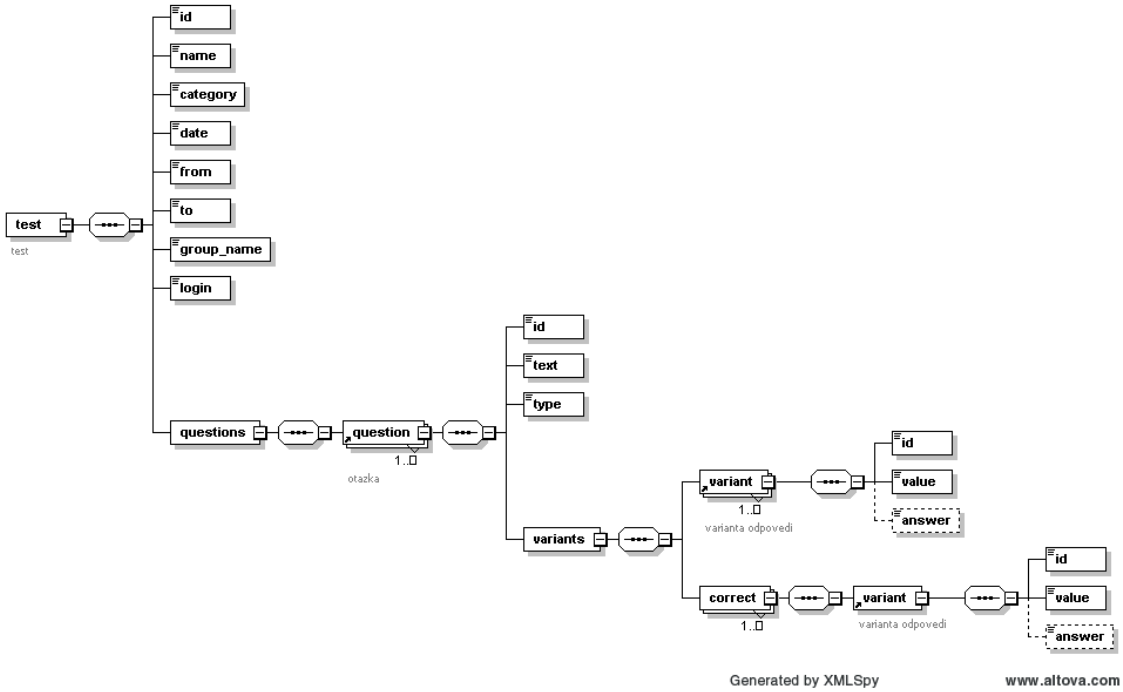
Obrázek 4.4: Databáze aplikace

- *type* - označení zda se jedná o otázku typu 1zN nebo NzN, čili zda je jedna správná odpověď nebo je více správných odpovědí na otázku.

Element *variant* obsahuje elementy:

- *id* - identifikátor otázky, využívá se jako určení pořadí odpovědi v rámci otázky.
- *value* - text odpovědi.
- *answer* - element obsahující informaci, že byla tato odpověď označena v testu studentem za správnou odpověď.

Elemet *correct* identifikuje správnou odpověď na otázku a pouze za označení této odpovědi je studentovi při vyhodnocování testu připsán bod za správnou odpověď.



Obrázek 4.5: Struktura XML

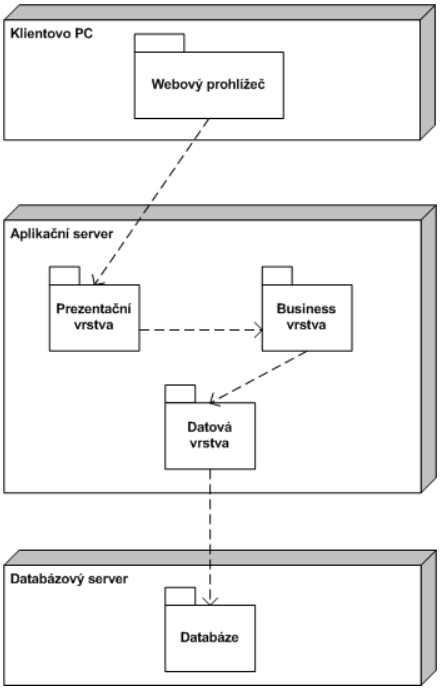
4.2.3 Architekrura systému

Aplikace je vyvíjena v ASP.NET, které je součástí .NET frameworku pro tvorbu webových aplikací. Pro přihlášení do aplikace lze využít jakýkoliv webový prohlížeč např. Internet Explorer, Operu, Firefox apod.

Architektura aplikace je rozdělena do tří vrstev, a to datové, business a prezentační vrstvy. Datová vrstva slouží pro přístup k datovému serveru a připojení k databázi. Business nebo také logická vrstva obsahuje především třídy reprezentující objekty domény z obr. 4.3. Logická vrstva zpracovává získaná data z databáze a z aplikace, k čemuž využívá uložených procedur na straně databázového serveru. Prezentační vrstva slouží k prezentaci dat a představuje samotné GUI aplikace.

4.2.4 GUI

Při tvorbě jakéhokoliv systému je zapotřebí zabezpečit jednotný vzhled aplikace z důvodu uživatelské přívětivosti systému. Pro zajištění jednotného vzhledu jednotlivých stránek lze



Obrázek 4.6: Architektura systému

v ASP.NET využít tzv. master page, která obsahuje definici chování a vyhledu společných částí všech stránek. Do master page lze taktéž vkládat ovládací prvky, které jsou součástí všech stránek. V případě této aplikace je v master page definováno záhlaví a zápatí stránek a popis vzhledu je definován v css souboru, na který se z master page odvolává viz kapitola 4.4.2. Ukázky praktického využití v ukázkové aplikaci viz obr. 4.9, 4.10, 4.11.

V aplikaci lze mít definovaných více master pages, a to v případě, kdy je zapotřebí zachovat základní vzhled aplikace stejný, ale na některých stránkách je zapotřebí skrýt vybrané prvky. Dalším případem použití může být zohlednění při definici vzhledu administrátorské a uživatelské části a přitom zachování stejného základního vzhledu, jako je např. barevné schéma apod. V ukázkové aplikaci je to např. přihlašovací stránka, kde se pod záhlavím nezobrazuje menu aplikace.

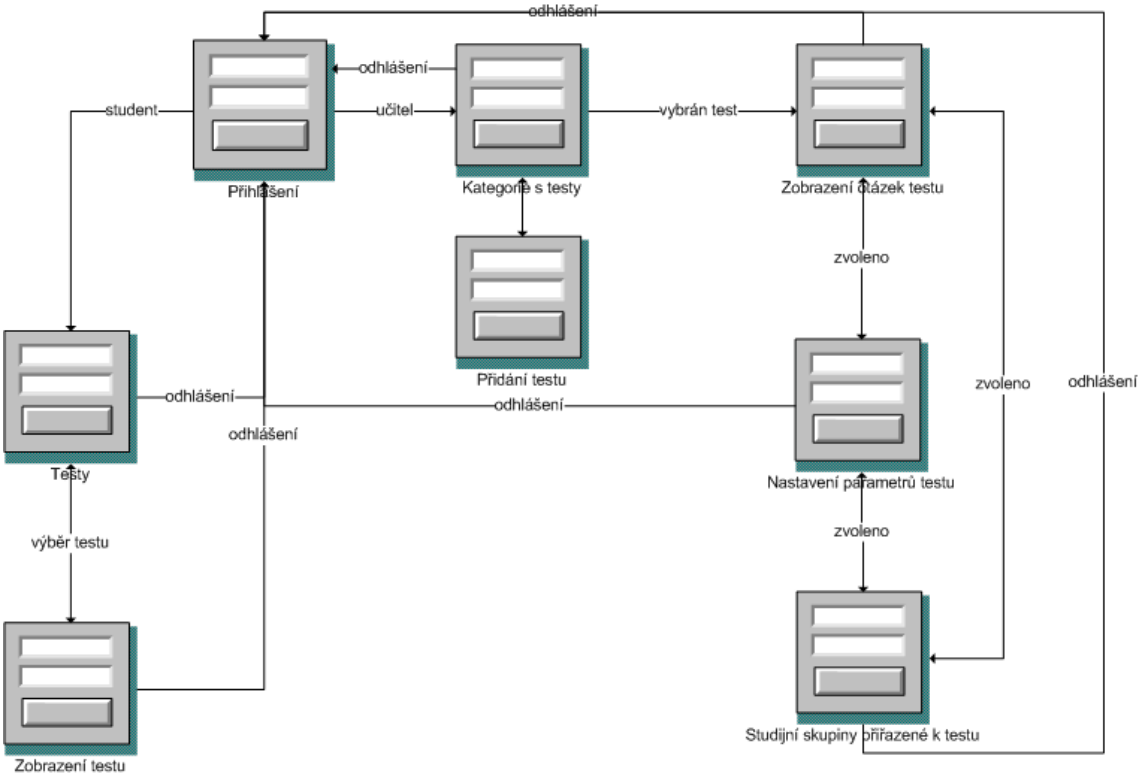
Návaznost jednotlivých stránek je rozdílná z hlediska přihlášeného uživatele a jeho role v systému. Diagram aplikace a její obrazovky jsou znázorněny na obrázku 4.7.

4.3 Použité technologie

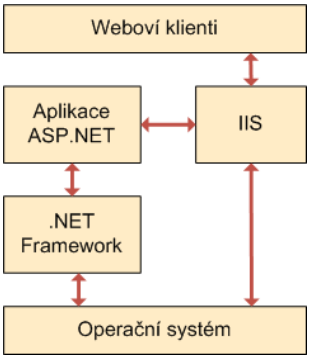
V rámci diplomové práce bylo využito při analýze i samotném vývoji aplikace více programů a technologií. Samotná ukázková aplikace je postavena na .NET frameworku 3.5.

4.3.1 ASP.NET

ASP.NET (součást .NET frameworku) je nástupcem ASP (Active server pages) a je určeno pro vývoj webových aplikací s služeb. Architektura technologie ASP.NET je znázorněna na obr. 4.8.



Obrázek 4.7: Diagram návaznosti jednotlivých stránek aplikace



Obrázek 4.8: Architektura technologie ASP.NET, převzato z [21]

ASP.NET jako součást .NET frameworku je taktéž založeno na CLR (Common Language Runtime), který je sdílen všemi aplikaci postavenými nad tímto frameworkem. Lze tak realizovat projekty v jazycích, které podporují CLR, jako jsou Visual Basic.NET, JSCript.NET, C# a mnoho dalších. ASP.NET aplikace jsou rychlejší z důvodu předkompilování do DLL souborů.

Pro práci s XML jsou dostupné knihovny v System.XML, v ukázkové aplikaci zejména třída XmlDocument. Seznam všech tříd viz [17].

4.3.2 T-SQL

T-SQL je proprietální rozšíření jazyka SQL od firmy Microsoft a Sybase. Jazyk SQL byl tímto rozšířen o transakční kontroly, zpracování výjimek a chyb, řádkové zpracování a deklarování proměnných. T-SQL podporují databázové servery MS SQL Server a Sybase SQL Server.

4.3.3 Nástroje

Pro vývoj aplikace byly použity Microsoft Visual Studio 2008, MS SQL Server 2008 a Altova XML Spy. Pro analýzu a návrh Altova XML Spy a Enterprise Architect. Všechny programy a technologie byly použity na operačním systému Windows XP Service Pack 2 a IIS server, který je jeho součástí.

Microsoft Visual Studio

Visual Studio představuje prostředí pro vývoj různých aplikací, jako jsou konzolové aplikace, GUI aplikace, WinForms aplikace, webové stránky, webové aplikace, webové služby ve strojovém kódu, ve spravovaném kódu na platformách Microsoft Windows, Windows Mobile, Windows CE, .NET, .NET Compact Framework a Microsoft Silverlight. Součástí Visual Studia je editor kódu, debugger, designer, možnosti průzkumníků a je možné přidávat různá dostupná rozšíření.

Prostřednictvím Visual Studia lze přímo přistupovat k databázi prostřednictvím data exploreru, lze vytvářet XML schémata a ostatní součásti projektu, jako jsou i css styly v rámci webových projektů.

MS SQL Server 2008

Podpora XML na straně databázového serveru je popsána v kapitole 3. Pro správu dat a jednotlivých databází běžících na MS SQL Serveru se používá Microsoft SQL Server Management Studio, které je jeho součástí. Management studio se používá i pro správu zabezpečení a nastavení celého databázového serveru.

Možnosti samotného MS SQL Serveru se odvíjí od jeho verze. V základní verzi se např. nenachází Business intelligence více viz [15] .

Altova XMLSpy

Altova XMLSpy je nejrozšířenější XML editor a vývojové prostředí pro modelování, opravu, transformaci a ladění XML technologií. Nabízí grafický návrh schématu, generátor kódu, konvertor souborů, debugger, analýzu, plnou podporu integrace databází, podporu pro XSLT, XPath, XQuery, WSDL, SOAP, XBRL a Office Open XML (OOXML) dokumentů, dále Visual Studio a Eclipse pluginů apod. [12]

XMLSpy má komplexní podporu pro práci s XML technologiemi prostřednictvím svého intuitivního uživatelského rozhraní a bohaté palety různých pohledů a možností editace XML. Pět synchronizovaných zobrazení pro editaci XML umožňuje pracovat s technologiemi XML a to způsobem, který nejlépe vyhovuje uživateli a danému dokumentu. Lze editovat XML v grafické podobě pomocí vkládání jednotlivých elementů, podobně jako v jiných návrhových systémech, taktéž lze editovat i textovou podobu dokumentu. Mezi jednotlivými pohledy na dokument je možno se přepínat a změny provedené v jednom z náhledů se automaticky promítnou i v ostatních pohledech.

V diplomové práci bylo XMLSpy použito pro návrh XML schématu testu, příklady výstupu viz 4.5 a příloha C.

Enterprise Architect (EA)
Enterprise Architect je dílem firmy Sparx Systems a představuje UML nástroj pro pokročilé modelování a návrh systémů. EA podporuje UML verze 2.1 a nabízí výkonné prostředky pro obchodní modelování, systémové inženýrství, návrh podnikových procesů, správu požadavků, software design, generování kódu, testování a další. Prostřednictvím EA lze modelovat celý životní cyklus aplikace od sběru požadavků až po nasazení systému. Více viz [20].

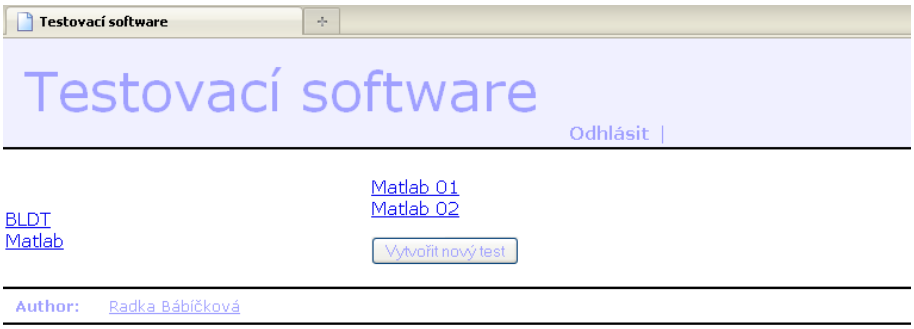
EA bylo v diplomové práci použito pro návrh struktury databáze obr. 4.4 a návrh use case diagramů obr. 4.1 a obr. 4.2.

4.4 Realizace a ukázky použití

4.4.1 Aplikace

Aplikace má jedno přihlašovací okno viz obr. 4.9 a na základě přidělené role uživateli se po přihlášení aplikace dělí na studentskou část viz obr. 4.11 a část pro učitele viz obr. 4.10.

Obrázek 4.9: Přihlašovací obrazovka

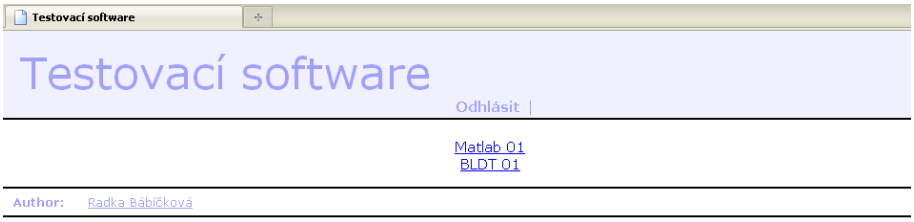


Obrázek 4.10: Zobrazení kategorií a příslušných testů vybrané kategorie

4.4.2 Ukázka Master Page

Ukázka login.Master stránky:

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="login.master.cs"
```



Obrázek 4.11: Aktuální testy studenta

```
Inherits="Tests.Web.Pages.MasterPages.login" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Testovací software</title>
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
    <link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="form1" runat="server">
  <div>
      <div id="header">
        <span class="headerTitle">Testovací software</span>
      </div>
    <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
    <p>
    </p>
    </asp:ContentPlaceHolder>
      <div id="footer">
        <div>
        <strong>Author: </strong>
        <a class="footerCol2" href="mailto:r.babickova{insert@here}gmail.com"
        title="Email author">Radka Bábíčková</a>
        </div>
      </div>
   </div>
   </form>
</body>
</html>
```

Vazba na Master Page v kódu samotných stránek:

```
<%@ Page Language="C#" MasterPageFile="~/Pages/MasterPages/login.Master"
AutoEventWireup="true" CodeBehind="Login.aspx.cs" Inherits="Tests.Web.Pages.
Login.Login" Title="Untitled Page" %>
```

4.4.3 Uložené procedury

Při práci s daty uloženými v databázi systému se využívá uložených procedur prostřednictvím, kterých jsou data zpracovávány.

```
USE [testsw]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author: Radka Bábíčková
-- Description: ziskani testu, na ktere muze student odpovidat
-- =============================================
ALTER PROCEDURE [dbo].[tsStudentAktTestGet]
@UserId INT
AS
BEGIN
  -- SET NOCOUNT ON added to prevent extra result sets from
  -- interfering with SELECT statements.
  SET NOCOUNT ON;

  -- Insert statements for procedure here
  SELECT
  * from
 (select
  test.id as id,
  test.value (N'(/test/name)[1]', 'nvarchar(255)') as test,
  test.value (N'(/test/date)[1]', 'date') as den,
  test.value (N'(/test/from)[1]', 'time') as od,
  test.value(N'(/test/to)[1]', 'time') as do,
  sg.name as skupina,
  u.login
  from Test inner join StudyGroup as sg on Test.studygroup_id=sg.id
  inner join User_StudyGroup as usg on sg.id=usg.studygroup_id
  inner join "User" as u on usg.user_id=u.id
  where u.id=@UserId) as tb1
  where den=CAST(GETDATE() as date) and
  od<=cast(GETDATE() as time) and
  do>cast(GETDATE() as time)
END
```

Pro získávání dat z XML datového typu je využit jazyk XQuery, jeho metody a metody jazyka T-SQL, jako jsou např.:

```
metoda value(XQuery, SQLType):

test.value(N'(/test/name)[1]', 'nvarchar(255)') as test
```

```
metoda exist(XQuery):

answer.exist('/correct') as correct
```

Kapitola 5

Závěr

Diplomová práce je zaměřená na podporu XML na straně databázového serveru MS SQL Server 2008 a porovnání této podpory s jinými databázovými systémy. Podporu XML na straně datbázového serveru jsem si vyzkoušela prakticky, v rámci ukázkové aplikace nebo i na příkladech použitých v teoretické části diplomové práce.

MS SQL Server 2008 má dobrou podporu pro práci s XML, jak už po stránce uložení dat v XML formátu nebo manipulaci s těmito daty. Oproti databázovému serveru PosgreSQL má výbornou podporu pro práci a uložení XML, protože PostgreSQL je v začátcích vývoje této podpory. PostgreSQL má základní podporu XML datového typu a práci s ním, ale nemá plnou podporu ANSI SQL 2003 a taktéž podporu lze dále vylepšovat, např. možnosti indexování, XML schémata nebo podporu XQuery datového modelu.

Co se týče MS SQL Serveru a Oracle, tak je to mnohem složitější, jelikož oba databázové servery mají plnou podporu XML datového typu a manipulace s XML daty, jen každý ze systémů přistupuje k této podpoře jiným způsobem, co se týče vnitřního uspopřádání v DB serveru. Navenek pro uživatele není až tak podstatné, jak je to uspořádané uvnitř, pokud mu oba servery poskytují stejnou podporu a komfortní práci s XML datovým typem. Dle mého názoru jsou Oracle a MS SQL Server v podpoře XML na obdobné úrovni a při výběru rozhodují jiné preference, který server použít. Ze zkušeností z praxe vím, že MS SQL Server se používá při středně velkých aplikacích a při použití Microsoft technologií, na druhou stranu Oracle se využívá převážně při velkých systémech a s použitím technologií mimo Microsoft. Samotný Oracle sice má podporu .NET, ale i přesto se s .NET využívá spíše MS SQL Server. Velkou výhodou Oraclu je jeho nezávislost na operačním systému, zatímco u MS SQL Serveru je to stále spojeno s nemalými problémy.

Na diplomovou práci lze navázat dalším zkoumáním MS SQL Serveru 2008 a Oracle 11g z hlediska další podpory, bezpečnosti, výkonnosti, škálovatelnosti, podpory vývojářů, business intelligence, apod.

V rámci ukázkové aplikace byl prakticky využit XML datový typ, jazyk XQuery s rozšířením XML DML, XML schéma na straně MS SQL Serveru. V rámci diplomové práce byla vytvořená ukázková aplikace, která umožňuje učitelovi vidět kategorie testů, ke kterým má oprávnění. V rámci těchto kategorií má možnost spravovat příslušné testy. Testu lze odebírat nebo přidávat otázky, nastavovat parametry, přiřadit nebo odebrat studijní skupinu. Učitel dále může vytvářet i nové testy, kdy nejdříve vytvoří test se základním nastavením a až následně může přidávat otázky a studijní skupinu. Ukázková aplikace má i část pro studenty, kde se po přihlášení studentovi zobrazí aktuální testy, které může vyplňovat. Po vyplnění student test uloží a ten je vyhodnocen a studentovi jsou zobrazeny výsledky testu.

Ukázkovou aplikaci lze rozšířit a doplnit, tak aby byla použitelná v produkčním prostředí.

Zejména by bylo zapotřebí doplnit některé validace, funkce typu automatického generování testů, rozšířit administrátorskou část, tak aby si sám učitel mohl vytvářet studijní skupiny, do kterých by si zařazoval studenty dle svých potřeb. Dále by bylo možné do aplikace doplnit přímý tisk vytvořeného testu, export testu do různých formátů, import sady otázek pro danou kategorii testu, případně aplikaci rozšířit o možnosti statistik zodpovězených testů.

Literatura

[1] Benoit, M.: XML v příkladech. Praha, Computer press 2000

[2] Bourret, R.: XML and Databases, [leden 2010], Dostupné z WWW: <http://www.rpbourret.com/>

[3] Coles, M.: Pro SQL Server 2008 XML. USA, Apress 2008

[4] Kosek, J.: XML schémata, [květen 2010], Dostupné z WWW: <http://www.kosek.cz/xml/schema/>

[5] Kosek, J.: Praktické využití schémat, [květen 2010], Dostupné z WWW: <http://www.kosek.cz/xml/devcon2003/foilgrp07.html>

[6] Kosek, J.: XQuery, [květen 2010], Dostupné z WWW: <http://www.kosek.cz/xml/2005devcon/>

[7] Lacko, Ľ: Microsoft SQL Server 2008 praktický sprievodca novinkami, [květen 2010], Dostupné z WWW: <http://msdn.microsoft.com/cs-cz/dd727769.aspx>

[8] Rydval, S.: Microsoft SQL Server 2000, [květen 2010], Dostupné z WWW: <http://www.rydval.cz/phprs/view.php?cisloclanku=2005123146>

[9] Samokhvalov, N.: XML Support in PostgreSQL, [květen 2010], Dostupné z WWW: <http://www.scribd.com/doc/4846375/XML-Support-in-PostgreSQL>

[10] Skonard, A.: Understanding XML Schema, [květen 2010], Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/aa468557.aspx>

[11] Sobotka, J.: Podpora pro práci s XML u databázového serveru Oracle. Brno, FIT VUT v Brně 2006

[12] Altova XMLSpy, [květen 2010], <http://www.altova.com/xmlspy.html>

[13] Insidsqlserver: The Evolution of Microsoft SQL Server: 1989 to 2000, [květen 2010], Dostupné z WWW: <http://insidesqlserver.com/companion/History of SQL Server.pdf>

[14] Microsoft: Microsoft SQL server 2008, [květen 2010], Dostupné z WWW: <http://www.microsoft.com/sqlserver/2008/en/us/default.aspx>

[15] MS SQL Server 2008 edice, [květen 2010], Dostupné z WWW: <http://www.microsoft.com/cze/sqlserver2008/editions.mspx>

[16] Microsoft msdn: SQL Server developer center, [květen 2010], Dostupné z WWW: <http://msdn.microsoft.com/en-us/sqlserver/default.aspx>

[17] Microsoft msdn: System.Xml Namespace, [květen 2010], Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/system.xml(VS.71).aspx>

[18] Oracle: Documentation Library, [leden 2010], Dostupné z WWW: <http://www.oracle.com/pls/db102/homepage>

[19] PostgreSQL: PostgreSQL 8.4.2 Documentation, [leden 2010], Dostupné z WWW: <http://www.postgresql.org/docs/8.4/interactive/index.html>

[20] Sparx Systems: Enterprise Architect, [květen 2010], Dostupné z WWW: <http://www.sparxsystems.com.au/products/ea/index.html>

[21] Technet: Architektura technologie ASP.NET, [květen 2010], Dostupné z WWW: <http://technet.microsoft.com/cs-cz/library/cc737863(WS.10).aspx>

[22] W3C Consortium: Extensible Markup Language (XML), [květen 2010], Dostupné z WWW: <http://www.w3.org/XML/>

[23] W3C Consortium: XQuery 1.0 and XPath 2.0 Data Model (XDM), [květen 2010], Dostupné z WWW: <http://www.w3.org/TR/xpath-datamodel/>

[24] XSQL - Combining XML and SQL, [leden 2010], Dostupné z WWW: <http://xsql.sourceforge.net/>

Seznam obrázků

Seznam použitých zkratek

- **ACL** - Access Control List - seznam přístupových práv
- **ASP** - Active server pages
- **BLOB** - Binary Large OBject - datový typ
- **CLR** - Common Language Runtime
- **COM** - Component Object Model - standard určující základní vlastnosti objektů a pravidla pro práci s nimi
- **DML** - Data Modification Language - jazyk pro modifikaci dat
- **DTD** - Document Type Definition - popis struktury nebo typu dokumentu
- **FLWOR** - For-Let-Where-Order by-Return - typ výrazu jazyka XQuery
- **EA** - Enterprise Architect
- **HTML** - HyperText Markup Language - značkovací jazyk pro webové stránky
- **IIS** - Internet Information Services - kolekce programů umožňujících realizovat HTTP, FTP servery.
- **LOB** - Large OBject - datový typ rozsáhlých dat
- **MS** - Microsoft
- **OLEDB** - Object Linking and Embedding Database - základní technologie podporující univerzální přístup k datům
- **PSVI** - Post-Schema Validation Infoset - otypovaný abstraktní datový model dokumentu XML
- **PL/SQL** - Procedural Language/Structured Query Language - procedurální dotazovací jazyk
- **SOA** - Service Oriented Architecture - určuje hranice pro architekturu aplikace a nebo její části.
- **SOAP** - Simple Object Access Protocol
- **SQL** - Structured Query Language - dotazovací jazyk

- **SQLCLR** - SQL Common Language Runtime - technologie pro přístup k .NET frameworku prostřednictvím SQL Serveru
- **T-SQL** - Transact-SQL - jazyk pro práci s daty uloženými v MS SQL Serveru
- **URI** - Uniform Resource Identifier - přesná identifikace zdroje informací
- **URL** - Uniform Resource Locator - adresa zdroje na internetu
- **W3C** - World Wide Web Consorcium - mezinárodní konsorcium vyvýjející webové standardy
- **XDM** - XQuery and XPath Data Model - datový model XQuery, XPath a XSLT
- **XDR** - XML Data-Reduced - zjednodušená verze jazyka XML-Data
- **XHTML** - eXtensible hypertext markup language - značkovací jazyk pro webové stránky
- **XPath** - XML Path Language - dotazovací jazyk nad XML daty
- **XQuery** - XML Query Language - dotazovací jazyk nad XML daty
- **XML** - Extensible Markup Language - obecný značkovací jazyk
- **XSLT** - eXtensible Stylesheet Language Transformations - jazyk pro transformaci XML dat do jiného formátu dat

Příloha A

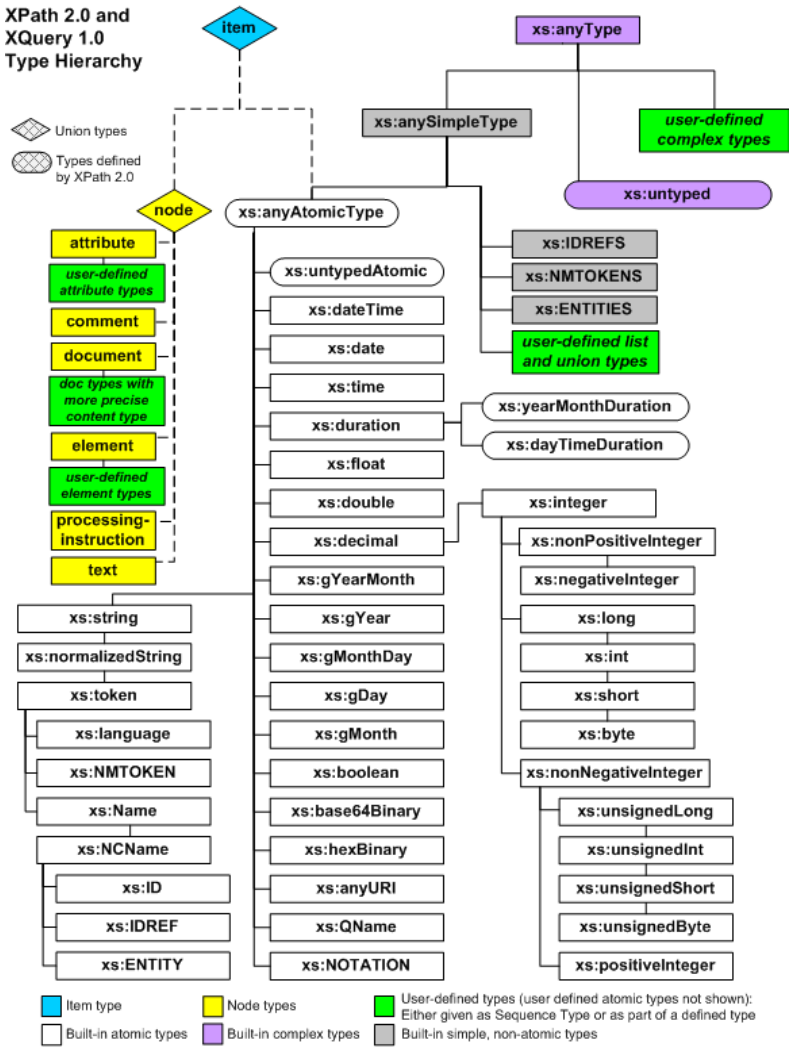
Obsah DVD

Součástí diplomové zprávy je přiložené DVD s následujícím adresářovým obsahem:

- Aplikace - adresář obsahující zdrojové kódy aplikace
- Dokumentace - obsahuje soubor ve formátu pdf
- Návrh aplikace - návrhové diagramy: Use case, Diagram tříd, DB diagram, Diagram aktivit, Architektura systému, Diagram obrazovek
- Nástroje - implementační nástroje Visual Studio 2008 Express, MS SQL Server 2008 Express, Altova XML SPY 2010, Enterprise Architect
- Zdrojove_texty_tex - zdrojové soubory dokumentace

Příloha B

Datové typy XPath a XQuery



Obrázek B.1: Datové typy XPath 2.0 a XQuery 1.0, převzato z [23]

Příloha C

XML schéma testu

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault=
"qualified" attributeFormDefault="unqualified">
  <xs:element name="test">
    <xs:annotation>
      <xs:documentation>test</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="id" type="xs:integer"/>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="category" type="xs:string"/>
        <xs:element name="date" type="xs:date"/>
        <xs:element name="from" type="xs:time"/>
        <xs:element name="to" type="xs:time"/>
        <xs:element name="group_name" type="xs:string"/>
        <xs:element name="login" type="xs:string"/>
        <xs:element name="questions">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="question" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
 </xs:element>
  <xs:element name="question">
    <xs:annotation>
      <xs:documentation>otazka</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="id" type="xs:integer"/>
        <xs:element name="text" type="xs:string"/>
```

```
        <xs:element name="type">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="1zN"/>
              <xs:enumeration value="NzN"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="variants">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="variant" maxOccurs="unbounded"/>
              <xs:element name="correct" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="variant"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="variant">
    <xs:annotation>
      <xs:documentation>varianta odpovedi</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="id" type="xs:integer"/>
        <xs:element name="value" type="xs:string"/>
        <xs:element name="answer" type="xs:boolean" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```